B00856253
CSCI 5408

Assignment 3

Cloud Setup Process

- Created an EC2 Instance in Amazon Web Service, with Ubuntu 16 OS. After that, created a secure key pair and successfully connected using Putty terminal as instructions given in Lab 5.
- Installed Spark 2.4.5 as per given instructions in Lab 5 and installed MongoDB 3.4 with required mongo-spark-connector [1].
- Configured instance's incoming ports to access Spark Master node's UI.
- Created master and slave node and specified 1 core for the worker node. If we do not specify the number of cores allowed for a worker node, otherwise it will use all the cores available. Since it is a free EC2 instance, it has only one core, so the worker node does not occupy all resources [2].

API Setup Process, Data Extraction Process and Cleaning Process

1. Twitter

- Created a Twitter Developer account as per given instructions in Lab 6.
- Referred official documentation of Twitter API to explore the search and streaming tweets, which returns data in JSON format [3].
- Wrote a python script, which uses packages "tweepy" along with "re", "pandas" to process the fetched tweets [4].
    - Using cursor of tweepy, created "api.search" call and specified keywords as "Canada OR University OR Dalhousie University OR Canada Education OR Halifax" to tweets containing one of these keywords.
    - Fetched 6000 tweets and extracted "full_text" attribute along with other fields like "location", "time", "reteet_count", etc.
    - "full_texr" column from fetched data is then cleaned using regular expression ("re" package) and substring matching.
    - Create a regular expression for matching and deleting URLs, emoticons, punctuation from fetched data.
- Using "pymongo", this cleaned data directly uploaded into EC2 instance's MongoDB server. For this, the MongoDB server must be unbound to localhost, and incoming port 27017 should be open. Then create a user in MongoDB with credentials, to upload data directly to the cloud instance [6].
- After Removing special characters, emoticons, URLs while retaining "RT" from "full_text", uploaded the data into EC2 instance's MongoDB server.

2. News Articles:

- Created a developer account and obtained API key from given website (https://newsapi.org).
- Defined provided search keyword as parameters in API call and fetched result.
- Converted result into JSON format and cleaned "content" column as did in tweet cleaning.
- This cleaned data again directly uploaded to instance's MongoDB server using "pymongo".

3. Movie Data

- Created a developer account and obtained API key from given website (www.omdbapi.com).
- Created API call to search for given keywords.
    - Search by string – only gives limited details for each retrieved movie(no rating, plot, etc.)

  - Search by Title – gives all the information required for this assignment for a particular movie.
- Due to this limitation, first search by string to get all titles of movies and store them into the list. Using this list and search by title option, get plot, ratings, genre for every movie.
- Cleaned this data as done in tweet processing and uploaded to instance's MongoDB server directly using "pymongo".

Data Processing in Spark

- Started a master and worker with 1 computing core specification using following command
  - sudo /sbin/start-master.sh
  - sudo /sbin/start-slave.sh <Master URL> --cores 1
- started pyspark with spark-mongo-connector and master node address with MongoDB conf
- --conf option is used to make mongo connection alive for 2 minutes otherwise in while executing code, connection may close after default interval [7].
  - /bin/pyspark --packages org.mongodb.spark:mongo-spark-connector_2.11:2.4.1 --conf mongodb.keep_alive_ms=120000 --master <Master URL>
- After successfully loading all the required data from MongoDB, created "pyspark sql dataframe", that allows complex manipulation of data using built-in functions.
- Using this dataframe, split the data using space and converted into words, which are filtered according to keywords provided for wordcount.
- After this filtering wordcount done in 2 ways:
  1. RDD API: using map and reduce functions of pyspark [6].
  2. DataFrame: using pyspark sql dataframe's count function [5].
- Final word count is stored into MongoDB using write() and format() function of spark as Dataframe. Also created an "output.txt" file and stored the output as string in that file.

Sample Structures:

1.cleaned data loaded into MongoDB

```
> db.tweet.find().pretty().limit(1)
{
        "_id" : ObjectId("5e7e71805184lee2083201ba"),
        "text" : "rt hanyang universitys  future talents education center freshm
en oath from nuest amp seventeen jeonghan amp scoups\n\nwelcome scoups amp jeong
han\npledis seventeen\nhttpstcotbfzfelz",
        "user_location" : "",
        "retweet_count" : 661
}
```

2. Wordcount output store into MongoDB

```
> db.out.find().pretty().limit(1)
{
        "_id" : ObjectId("5e7e75e3dd37486f061adbe8"),
        "words" : "computer science",
        "count" : NumberLong(3)
}
>
```

Reference:

[1] Hevo Blog - Transformative ideas and real insights on all things Data. 2020. *Install Mongodb On Ubuntu 16.04 LTS - Hevo Blog*. [online] Available at: <https://hevodata.com/blog/install-mongodb-on-ubuntu/> [Accessed 27 March 2020].

[2] Applications, W., 2020. *Why Do We Need To Specify Executor Cores For Spark Applications?*. [online] Community.cloudera.com. Available at: <https://community.cloudera.com/t5/Support-Questions/Why-do-we-need-to-specify-executor-cores-for-Spark/td-p/235306> [Accessed 27 March 2020].

[3] Docs. (2020). Retrieved 27 March 2020, from https://developer.twitter.com/en/docs

[4] Docs.python.org. 2020. *Re — Regular Expression Operations — Python 3.8.2 Documentation*. [online] Available at: <https://docs.python.org/3/library/re.html> [Accessed 27 March 2020].

[5] GitHub. 2020. *Apache/Spark*. [online] Available at: <https://github.com/apache/spark/tree/master/examples/src/main/python> [Accessed 27 March 2020].

[6] Api.mongodb.com. 2020. *Pymongo 3.9.0 Documentation — Pymongo 3.9.0 Documentation*. [online] Available at: <https://api.mongodb.com/python/current/> [Accessed 27 March 2020].

[7] Spark.apache.org. 2020. *Spark Standalone Mode - Spark 2.4.5 Documentation*. [online] Available at: <https://spark.apache.org/docs/latest/spark-standalone.html> [Accessed 27 March 2020].

[8] Spark.apache.org. 2020. *Examples | Apache Spark*. [online] Available at: <https://spark.apache.org/examples.html> [Accessed 27 March 2020].