DIC LAB II REPORT

asishkak-50288695 | swapnika-50289464

Objective: The main goal of the project is to aggregate data from multiple sources and analyze the data using big data method of MapReduce and finally storing the data collected on WORM infrastructure Hadoop. A visualization data product is built from this analysis.

Procedure:

Development language used: Python

Topic of Interest: Politics

Sub-topics: Mueller Report, Border Wall, Trump, North Korea – USA Summit, Trump retreats on Healthcare

STEP-1:DATA COLLECTION

Data is collected from 3 sources which include twitter, NYTimes and Common Crawl.

Twitter:

- First, we get the API keys from the twitter developers website that are used to get the data.
- These keys are given OAuthHandler for authentication, just to make sure that the right person with right keys is accessing the data. The below lines explain it.

```
auth = tweepy.OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token,access_secret)

api = tweepy.API(auth)
```

- Now, with the necessary keywords, we have collected tweets since 2019-01-01 which are originated within USA by using '39.8,-95.583068847656,2500km' as geocoordinates with radius.
- We wrote all the retrieved tweets to a text file by filtering out retweets by post fixing *"-filter: retweets*" to the keywords.
- Various arguments include query, since, geo-location etc.,

NYTimes:

- API keys are obtained from NYTimes and these keys are used to collect articles.
- The url's are collected by passing the arguments query, begin date, end date and pages range. This is done by using "*articleAPI('key').search()".*
- From the list of URLs, the articles are obtained using BeautifulSoup.

Common Crawl:

- Common Crawl is a public archive which maintains snapshots of crawled data for various domains.
- In this assignment, we have to collect the most recent data since Jan 2019, and which also should be related to our keywords. So, the snapshots since 2019 are 2019-13, 2019-09, 2019-04.
- Now, the process of collecting common crawl is not by using API keys but a bit different.
- Common Crawl stores data as archives like WARC, WET (Text Archives), indexes etc.,
- Now, to retrieve data from WARC, first we should know where our data from a certain domain is located in the snapshots.
- So, we use *http://index.commoncrawl.org/* to get a JSON response of records where each record has its index, offsets, and length of data we want to retrieve.
- By using these indices, offset and length. We crawl the data using BeautifulSoup and filtering out all the <p> tags in the webpage.
- After retrieving the text, we now write the data to a text file.

STEP-2: PROCESSING OF DATA

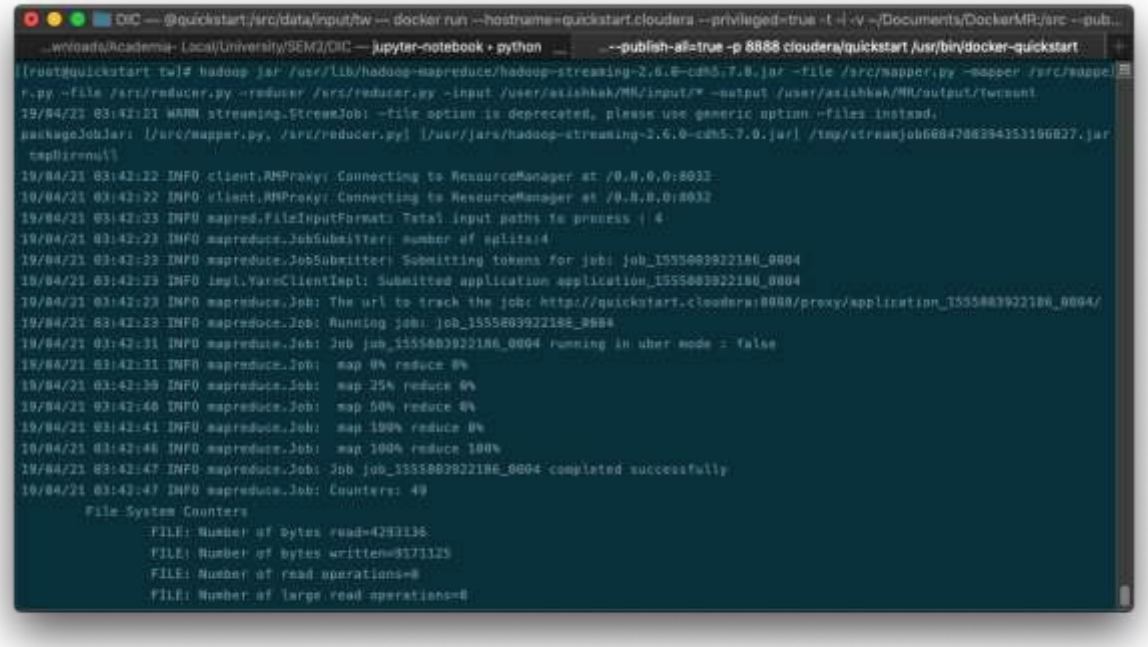
- In this step, we remove all the unnecessary junk in our corpus.
- In data pre-processing we first make our entire corpus to lowercase. So, when we try to remove stop words in the later steps. We won't have any case-sensitive problems.
- In the next step links and usernames are removed with the help of regex library.
- Special characters and symbols are deleted with the help of a string matching and also, we've included one more step to remove all the left out special characters and numbers by removing all other than "\n" and alphabets by using regex library.
- Finally, stemming is done to reduce the words in our corpus to its stem words. So, when we reduce them to stem words. It becomes easy and the probability of achieving a higher word count is very high. This is done by using SnowballStemmer from nltk library.
- Code: *preprocessing.ipynb*

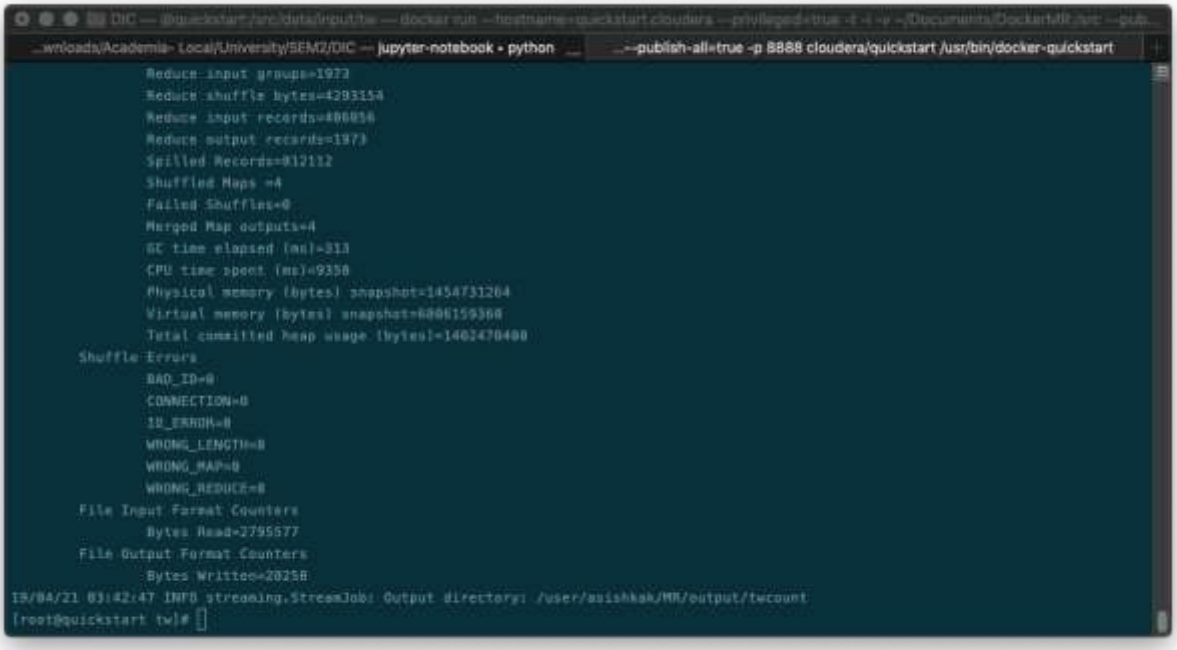
STEP-3: BIG DATA INFRASTRUCTURE

- Hadoop infrastructure is set up for storing and running MapReduce jobs on the collected corpus.
- This is done using the cloudera docker image which is provided to us.
- Now, we start the docker image using the below command.
    - *docker run –hostname=quickstart.cloudera –privileged=true -t -i -v /Users/Asish/Documents/DockerMR:/src –publish-all=true -p 8888 cloudera/quickstart /usr/bin/docker-quickstart*
        - I have provided */Users/Asish/Documents/DockerMR:/* as my *src* folder because dockers has a specific set of directories which are mounted to docker containers. In my case,
            - /Users
            - /Volumes

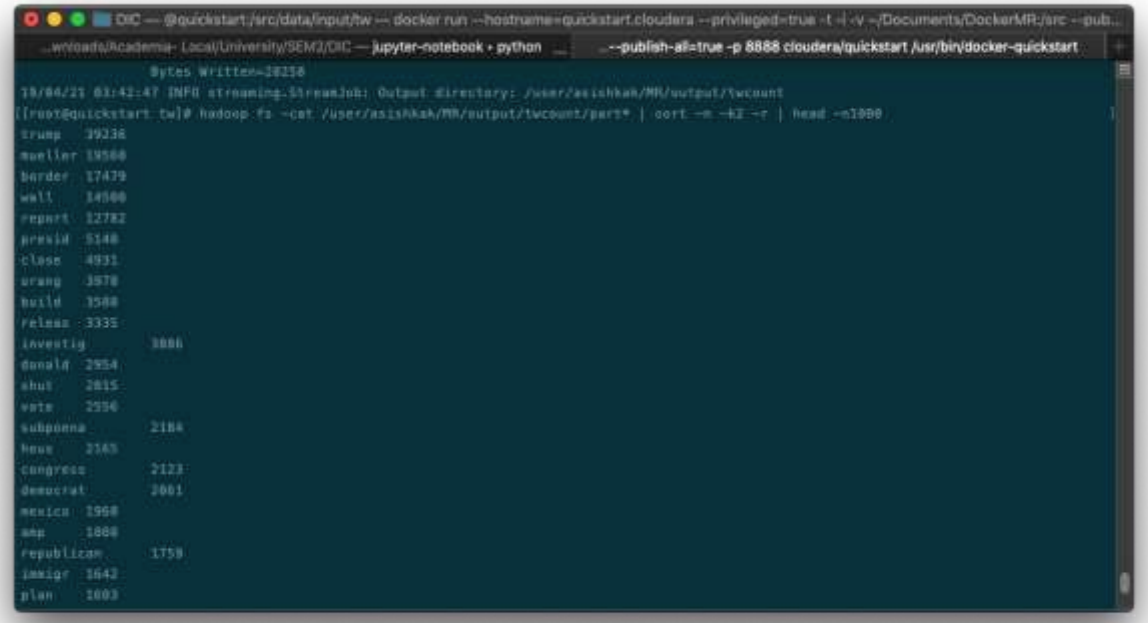
- /private
- /tmp

- Now, let's create few folders in Hadoop file system where we can put our corpus. To create, we use the below commands.
  - hadoop fs -mkdir /user/asishkak
  - hadoop fs -mkdir /user/asishkak/MR
  - hadoop fs -mkdir /user/asishkak/MR/input
  - hadoop fs -mkdir /user/asishkak/MR/input/twitter
  - hadoop fs -mkdir /user/asishkak/MR/input/nyt
  - hadoop fs -mkdir /user/asishkak/MR/input/cc
- Now, let's create few folders in Hadoop file system where we can get output. To create, we use the below commands.
  - hadoop fs -mkdir /user/asishkak/MR/output
- Now, we copy the corpus from our local file system to hadoop file system by using the below commands.
  - We move data to their specific folders.
  - For instance, twitter data is added to its specific folder in hadoop file system.
    - hadoop fs -put *.txt /user/asishkak/MR/input/twitter/
- Now, we've written MapReduce algorithms for both word count and word co-occurrence (Pair method).
- To execute them on our corpus, we give in mapper and reducer python files and input files on which mapping and reducing happens. The command to execute it is as below.
  - *hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.7.0.jar -file /src/mapper.py -mapper /src/mapper.py -file /src/reducer.py -reducer /src/reducer.py -input /user/asishkak/MR/input/twitter/* -output /user/asishkak/MR/output/twitter*





- A part_0000 is generated where we have out word count / word co-occurrence.
- To see the output. We execute the below command.
    - hadoop fs -cat /user/asishkak/MR/output/part*
- To view the output in sorted descending order of word count / co-occurrence.
    - hadoop fs -cat /user/asishkak/MR/output/part* | sort -n k2 -r



- To download the word count/ co-occurrence to our local file system. We can execute two commands which are
    - Hadoop fs -get /user/asishkak/MR/output/twitter/*
    - hadoop fs -cat /user/asishkak/MR/output/part* | sort -n k2 -r >> filename.txt

STEP-4: VISUALIZATION AND ANALYSIS

- In this step, we've used tableau for visualization.
- We have used the output file (text file) which contains the counts of words/ word co-occurrences respectively to generate a word cloud.
- For Instance, the word cloud generated for top 10 and 100 words for twitter corpus is below.