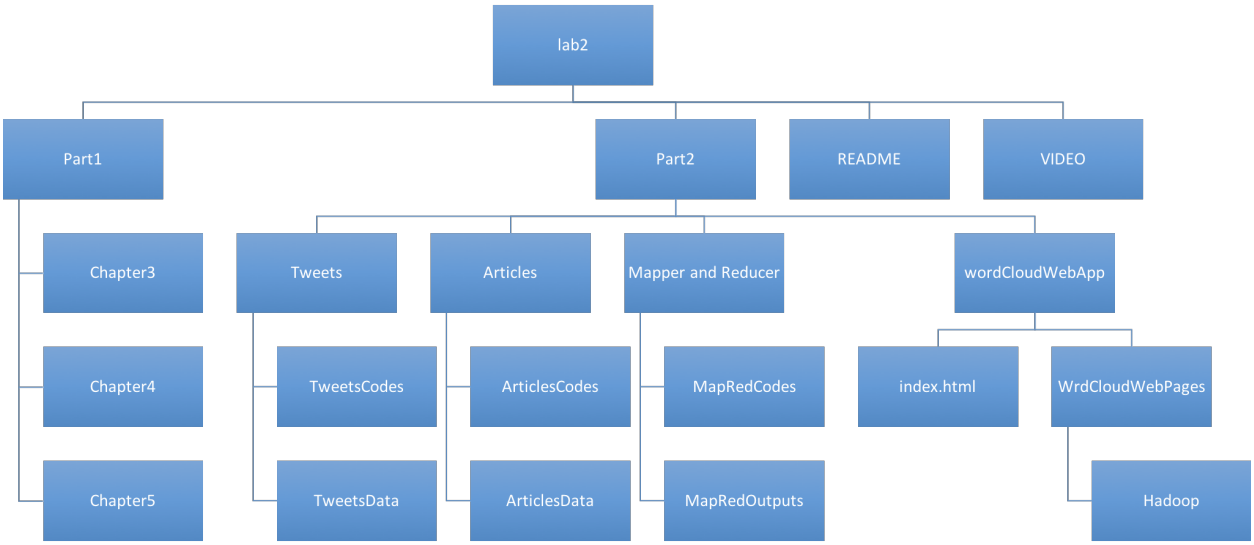
CSE4/587 DATA-INTENSIVE COMPUTING SPRING 2018

LAB2: DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

Parikshit Sunil Deshmukh - pdeshmuk - 50247649
Mahalakshmi Padma Sri Harsha Maddu - mmaddu - 50246769

Explanatory Video:
https://www.dropbox.com/s/new9eguqd1z4dp4/part1.mov?dl=0
https://www.dropbox.com/s/8szl6ph06gtdfcn/part2_wordCloud.mov?dl=0

Directory Structure:



Part1:
A number of basic python commands are tested and familiarized. The ipynb files are located in the above shown directory.

Part2:

Step1:
Key word for Search: "Spacex OR Tesla OR ElonMusk"

Collecting data from Twitter

- Tweets from Twitter API are collected using the key word.
- All of the following steps are repeated for 1 day of tweets and week's tweets.
- Removed the retweets.

- Extracted ‘text’ column from the tweets dataframe.
- Special Characters are removed from the tweets dataframe.
- All words of length ‘1’ are removed.
- All tweets are converted into lower case characters.
- The tweets are written into a ‘txt’ file.

Collecting data from NYTimes:

- Articles from NYTines API are collected using key word.
- All of the following steps are repeated for 1 day of articles and week’s articles.
- All the video are removed from the article unicode data.
- The webpage is scraped off the unnecessary tags and is converted to a string.
- Special Characters are removed from the articles.
- All words of length ‘1’ are removed.
- All articles are converted into lower case characters.
- The articles are written into a ‘txt’ file.

Output of Step1:
We now have the following data files.

- FinalTweets.txt,
- FinalArticles.txt,
- FinalTweets1Day.txt,
- FinalArticles1Day.txt

Step2:
Installation of Hadoop.

- Hadoop is installed using Oracle Virtual Box.
- Basic commands are tested for the sample data available.
- The following code snippet
    - Starts Hadoop:
        - start-hadoop.sh
    - Creates a directory in the dhfs:
        - hdfs dfs –mkdir user/hadoop/input
    - Moves folder from Hadoop-home directory to hdfs directory:
        - hdfs dfs -put /home/hadoop/books /user/hadoop/input
    - Run Word Count:
        - HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar wordcount input output
    - View Output files:
        - hdfs dfs -cat output/*
    - Stop Hadoop:
        - stop-hadoop.sh

Step3:
Loading the data into HDFS:

- Start Hadoop:
    - start-hadoop.sh
- Create two folders called “twitterData” and “newsData”
    - hdfs dfs -mkdir /user/hadoop/twitterData
    - hdfs dfs -mkdir /user/hadoop/newsData
- Push input files from local directory to hdfs(into the created input_file):
    - hdfs dfs -put /home/hadoop/lab2/FinalTweets1Day.txt /user/hadoop/twitterData
    - hdfs dfs -put /home/hadoop/lab2/FinalTweets.txt /user/hadoop/twitterData
    - hdfs dfs -put /home/hadoop/lab2/FinalArticles1Day.txt /user/hadoop/newsData
    - hdfs dfs -put /home/hadoop/lab2/FinalArticles.txt /user/hadoop/newsData

Step4:
Mappper and Reducer for 1 Day of NYTimes and Twitter Data: (mapperNorm and reducerNorm)

- mapperNorm and reducerNorm are responsible for MR processing of 1 day of data.
- Mapper prodcuces the the output in th afomrat <word, 1> and reducer then clubs it.
- We have obtained top 100 so words so that we can use them for word cloud

Now, the mapper and reducer python files are downloaded into the Hadoop home.

Executing MapReduce Wordcount:

- **Wordcount for 1 day’s tweets:**
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-*streaming*.jar
  -mapper "python /home/hadoop/lab2/mapperNorm.py"
  -reducer "python /home/hadoop/lab2/reducerNorm.py"
  -input /user/hadoop/twitterData/FinalTweets1Day.txt
  -output /user/hadoop/outputTweetsNorm1Day

- **Wordcount for 1 day’s Articles:**
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-*streaming*.jar
  -mapper "python /home/hadoop/lab2/mapperNorm.py"
  -reducer "python /home/hadoop/lab2/reducerNorm.py"
  -input /user/hadoop/newsData/FinalArticles1Day.txt
  -output /user/hadoop/outputArticlesNorm1Day

Step5:
Visualization using d3.js:

Go to: “wordcloudWebApp” -> index.html
Note: Open in Firefox for cross platform support

We have used D3 js framework for making word cloud to analyze the data collected. The output of Reducers which is a Tab separated text file is parsed in html page and the wordcloud is formed.

"wordcloudWebApp" ←this is the module that contains all the source code related to the WordCLoud formation.
Conclusion:
We can see from word cloud that terminologies related to Spacex like Tesla, Elon Musk, rocket are occurring most of the times.

Step6:
Repeating Step5 for a week's data of tweets and articles.
Executing MapReduce Wordcount:

- **Wordcount for a week's data of tweets.**
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar
  -mapper "python /home/hadoop/lab2/mapperNorm.py"
  -reducer "python /home/hadoop/lab2/reducerNorm.py"
  -input /user/hadoop/twitterData/FinalTweets.txt
  -output /user/hadoop/outputTweetsNorm

- **Wordcount for a week's data of Articles.**
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar
  -mapper "python /home/hadoop/lab2/mapperNorm.py"
  -reducer "python /home/hadoop/lab2/reducerNorm.py"
  -input /user/hadoop/newsData/FinalArticles.txt
  -output /user/hadoop/outputArticlesNorm

Step7:

Visualization of the output on a webpage **Go to:** "wordcloudWebApp" -> index.html

Step8: Co-occurrence

For Obtaining Co-occurrences we have implemented 3 different approaches for different kind of insights:

A. **Coding Mapper and Reducer Part: (Co-occurrence CoNew):**

=> In this approach we have first found the top ten words on the run in reducer and also in the same reducer we have implemented logic to get list of all co-occurrences in same article

and tweets for that particular word out of top 10 words.

Eg: Tesla : { elon, spacex, rocket, company}

Executing Map-reduce for WordCount (Co-occurrence CoNew) – 1 Day:

- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCoNew.py" -reducer "python /home/hadoop/lab2/reducerCoNew.py" -input /user/hadoop/twitterData/FinalTweets1Day.txt -output /user/hadoop/outputTweetsCoNew1Day
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCoNew.py" -reducer "python /home/hadoop/lab2/reducerCoNew.py" -input /user/hadoop/newsData/FinalArticles1Day.txt -output /user/hadoop/outputArticlesCoNew1Day

Executing Map-reduce for WordCount (Co-occurrence CoNew) – 1 Week:

- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCoNew.py" -reducer "python /home/hadoop/lab2/reducerCoNew.py" -input /user/hadoop/twitterData/FinalTweets.txt -output /user/hadoop/outputTweetsCoNew
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCoNew.py" -reducer "python /home/hadoop/lab2/reducerCoNew.py" -input /user/hadoop/newsData/FinalArticles.txt -output /user/hadoop/outputArticlesCoNew

A. Coding Mapper and Reducer Part: (Co-occurrence Final)

=> In this approach we have found the top 50 occurring words that is <word1, word2> , count. And then we have visualized them also.

Steps 5 and 6 .i.e., for 1 day of data and week's data, co-occurrence words are found.

Executing Map-reduce for WordCount (Co-occurrence Final) – 1 Day:

- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperFinal.py" -reducer "python /home/hadoop/lab2/reducerFinal.py" -input /user/hadoop/twitterData/FinalTweets1Day.txt -output /user/hadoop/outputTweetsFinal1Day
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperFinal.py" -reducer "python /home/hadoop/lab2/reducerFinal.py" -input /user/hadoop/newsData/FinalArticles1Day.txt -output

/user/hadoop/outputArticlesFinal1Day

Executing Map-reduce for WordCount (Co-occurrence Final) – 1 Week:

- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperFinal.py" -reducer "python /home/hadoop/lab2/reducerFinal.py" -input /user/hadoop/twitterData/FinalTweets.txt -output /user/hadoop/outputTweetsFinal
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperFinal.py" -reducer "python /home/hadoop/lab2/reducerFinal.py" -input /user/hadoop/newsData/FinalArticles.txt -output /user/hadoop/outputArticlesFinal

B. Coding Mapper and Reducer Part: (Co-occurrence Co)

=> In this approaches we have taken the list of already found top ten words from the very first mapreduce and then we have found the co-occurring words and sorted these phrases as in descending order. <word1 word2, count>

eg: Elon Musk, 190
rocket spacex, 100

Executing Map-reduce for WordCount (Co-occurrence Co) – 1 Day:

- Hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCo.py" -reducer "python /home/hadoop/lab2/reducerCo.py" -input /user/hadoop/twitterData/FinalTweets1Day.txt -output /user/hadoop/outputTweetsCo1Day
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCo.py" -reducer "python /home/hadoop/lab2/reducerCo.py" -input /user/hadoop/newsData/FinalArticles1Day.txt -output /user/hadoop/outputArticlesCo1Day

Executing Map-reduce for WordCount (Co-occurrence Co) – 1 Week:

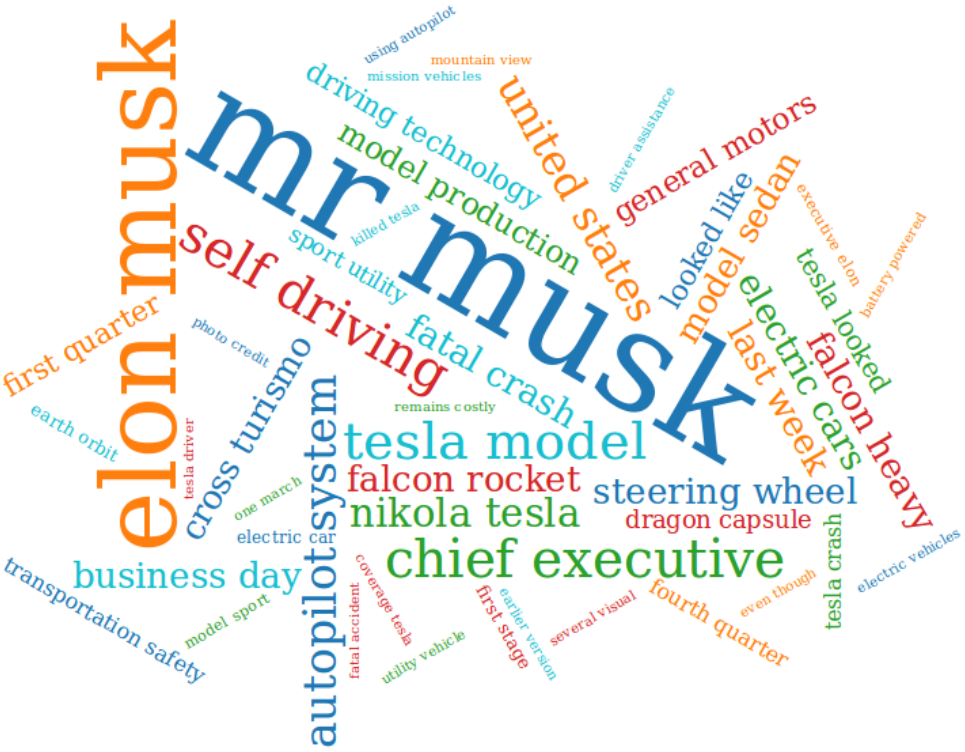
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCo.py" -reducer "python /home/hadoop/lab2/reducerCo.py" -input /user/hadoop/twitterData/FinalTweets.txt -output /user/hadoop/outputTweetsCo
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming.jar -mapper "python /home/hadoop/lab2/mapperCo.py" -reducer "python /home/hadoop/lab2/reducerCo.py" -input /user/hadoop/newsData/FinalArticles.txt

-output /user/hadoop/outputArticlesCo

Summary and Conclusion:

- After performing big data analysis on 1 day data form NYTimes and Twitter on SpaceX and Tesla we can see a big convergence by analyzing the respective word clouds. This denotes that our MapReduce has perfectly processed the data to gather trending topics and phrases related to our t

- From the word cloud of co-occurring phrases we can see the trending and most occurred phrase Articles and Tweets both. And it is also converging very well with the normal single word processed data.



- To improve the insight and for deep analysis we have used 1 week of big data for processing through Map-Reduce and have also formulated word cloud on this data, which also correlated fairly with the 1 Day data.

- From the Co-occurrences calculated in Part A of Co-occurrence section, we have found a list of co-occurring words that is in neighbour of the one top ten words. The output is as below:
  spacex, 68 set(['capsules', 'launch', 'launches', 'spacesuits', 'milestone', 'farhad', 'lists',

'officials', 'ambition', 'hopes', 'falcon', 'momentum'])

Referances:

- http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/
- https://github.com/jasondavies/d3-cloud
- https://github.com/wvengen/d3-wordcloud
- Google.com
- D3.js
- Bootstrap templates