Hadoop and Map-Reduce: Data Intensive Computing

Varun Nagaraj: 50290761

April 19th, 2019

1 Introduction

In this lab, we will expand our skills in data exploration developed in Lab1 and enhance them by adding big data analytics and visualization skills. Automate data collection from multiple sources using the APIs offered by the businesses, explain the importance of evaluating the reliability of data (for example: social media vs news media), and apply classical big data analytical methods: MapReduce for word count and related family of algorithms such as word occurrence and n-grams

2 Docker and Map-Reduce

Docker is used to run software packages called containers. Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.
I set up the docker ecosystem on my local machine by using the containers and images provided by Cloudera. Use the below command to get the latest docker container.

```
$ docker pull cloudera/quickstart:latest
```

2.1 Map-Reduce

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a map procedure (or method), which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a reduce method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). MapReduce is a framework for processing parallelizable problems across large datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware)

or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of the locality of data, processing it near the place it is stored in order to minimize communication overhead.

2.2 Word Count MR code

```
1  Mapper \newline
2  \newline
3  import sys \newline
4  for line in sys.stdin: \newline
5  words = line.strip().split() \newline
6  for word in words: \newline
7          print('\%s\t\%s' \% (word, 1)) \newline
8
9  \newline
10 Reducer \newline
11
12 import sys\newline
13 current\_word = None\newline
14 current\_count = 0\newline
15 word = None\newline
16 for line in sys.stdin:\newline
17     word, count = line.strip().split('\t', 1)\newline
18     try:\newline
19         count = int(count)\newline
20     except ValueError:\newline
21         continue\newline
22     if current\_word == word:\newline
23         current\_count += count\newline
24     else:\newline
25         if current\_word:\newline
26             print('\%s\t\%s' \% (current\_word, current\_count))\
                   newline
27         current\_count = count\newline
28         current\_word = word\newline
29 if current\_word == word:\newline
30     print('\%s\t\%s' \% (current\_word, current\_count)) \newline
```

2.3 MR run on Hadoop system

I ran the Mapper and Reducer in hadoop system by following the below steps.
1. Install the docker image on the system and checkout the latest version from Cloudera. After this start the docker container using the below command.

docker run –hostname=quickstart.cloudera –privileged=true -t -i -v /Users/xyz/Documents/SomeFolder:/src –publish-all=true -p 8888 cloudera/quickstart /usr/bin/dockerquickstart

2. After this is done, create a folder structure in the hadoop file system which is also known as HDFS.

hadoop fs -mkdir /user/Varun/MR/input

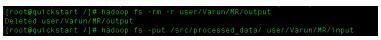
Create the above directory structure.
3. Add the files into the file system in hadoop using the below command.

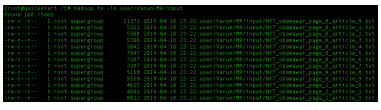
hadoop fs -put *.txt /user/Varun/MR/input

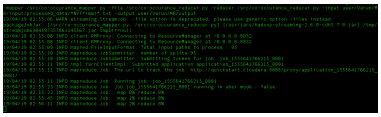
4. Run the word count mapper reducer using the hadoop streaming service jar file. The command is as shown below.

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/Varun/MR/input/* -output /user/Varun/MR/output

For Co-Occurrence programs change the mapper and reducer to the apt one.
Below are the images for the run of the above commands on Docker and Hadoop:

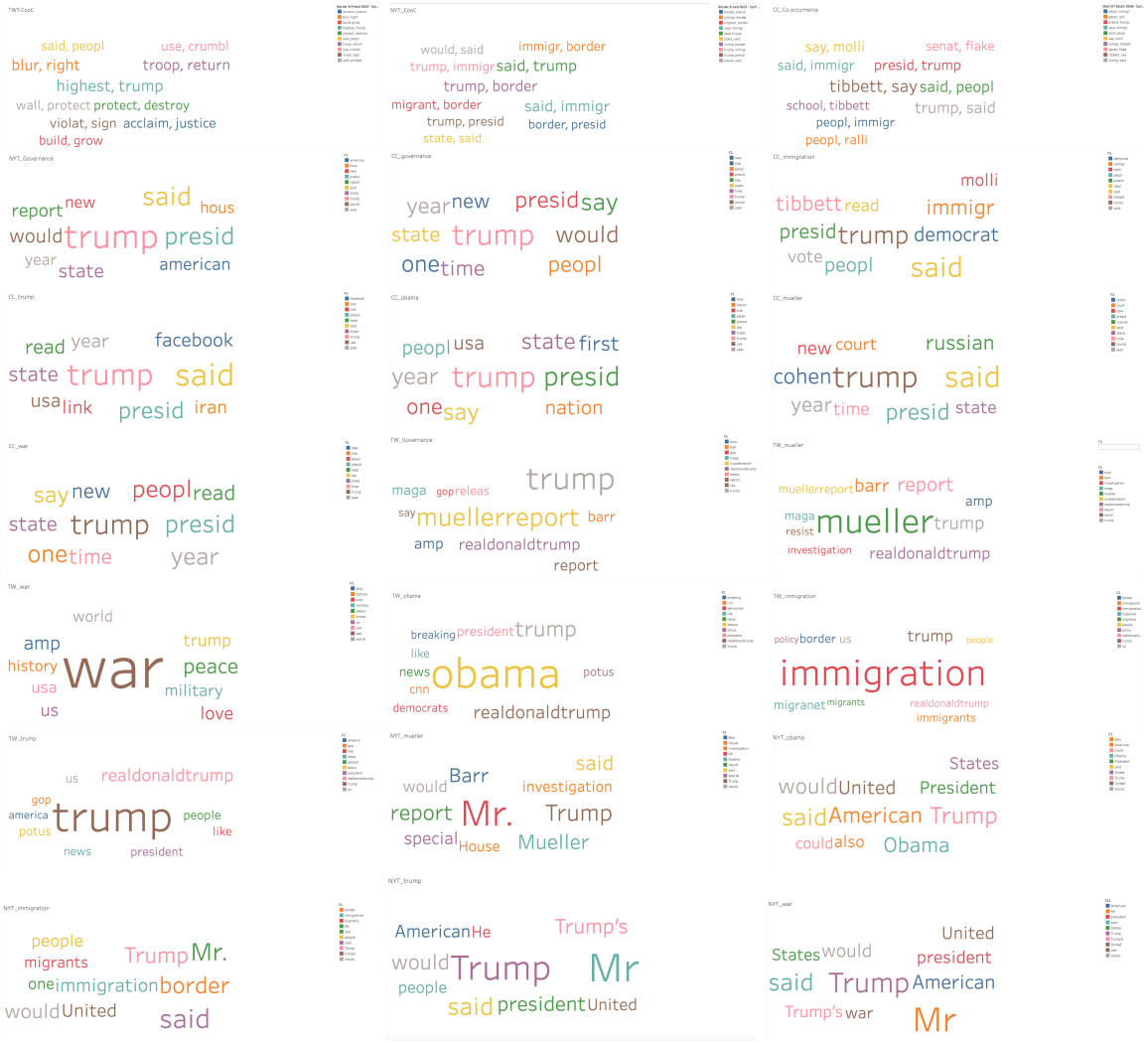






Outputs from docker.

3 Tableau Visualizations



Outputs of the Tableau Word Cloud

The Public repository can be found on Tableau Public:

https://public.tableau.com/profile/anudeep.shive.gowda#/vizhome/Word_Cloud$_{F}inal/NYT_{w}ar$