

# LAB 2: DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

Monisha Balaji, Roopa Chandrashekar  
University at Buffalo  
[mbalaji@buffalo.edu](mailto:mbalaji@buffalo.edu), [roopacha@buffalo.edu](mailto:roopacha@buffalo.edu)

## Abstract

This is the age of Big Data. The amount of data in our world has escalated to such high volumes in the last decade. Hence, critical efforts need to be structured for powerful usage of the available data. Big Data Analysis is thus the study of pervasive data to help recognize behavioral patterns. Data is generally categorized based on multiple parameters such as the high speed they are generated with, the huge volume they represent, the variety of information and the veracity. This project involves every individual to work on their data exploration skills. The computed data is exposed to data analytics and data visualization processing. This report records the working and results of the MapReduce word-count and word co-occurrence on the gathered data.

## 1. INTRODUCTION

Apache Hadoop provides the use of a selection of open-source software facilities that enable using many computers to manipulate the massive amounts of data to help with problem-solving. It provides the usage of one of the most popular software framework MapReduce is a computing paradigm for processing data, which aids distributed storage and processing of data. This project involves using Hadoop MapReduce structure to compute the word-count for the data collected and also execute word co-occurrence and retrieve the top 10 most frequently occurring words.

## 2. DATA COLLECTION

As a requisite for this lab, we are required to gather articles for New York Times, accumulate twitter data and perform web crawling to further gather more documents. It is required to collect data for any particular topic from any domain and also for five sub-topics for the chosen. The topic chosen for this project is “CRIME”. The sub-topics are “MURDER”, “RAPE”, “SMUGGLING”, “THEFT”, “TERRORISM”. The procedure followed to gather data for the respective categories are given below:

### 2.1 NYTIMES ARTICLE COLLECTION

New York Times news articles record the day-to-day happenings and make them available to the world. In order to be able to scrap the original articles, we require an NYTimes Developer account to acquire access to the APIs. This process also requires the installation and importing of packages such as ‘nytimesarticle’. This allows you to use the article API thus enabling you to assemble the required list of articles. It is essential to gather 100 articles for each of the headings chosen; a total summing to a minimum of 600 articles. Hitting the API returns JSON results. The next step involves extracting the article URLs from the ‘response’ field. Once the URL list is acquired, it is necessary to remove duplicate URLs from the list before data extraction. Now that we have the final URL list in our hands, we hit each URL from the list and pluck out the text field from the article.

## 2.2 TWITTER DATA COLLECTION

Twitter Data collection can be expanded from the procedure followed in the previous project. Here, we have used the `rtweet` package to help with Twitter data collection and processing. It allows the use of functions which after the authentication of your twitter key credentials, helps collect streaming tweets using the Twitter Search API. The procedure is as follows:

1. Used '`search_tweets()`' function to collect recent tweets. This is done using multiple keywords related to our topic 'CRIME'. One of the parameters we use while searching is `lookup_coords()` which makes use of the Google API key; it basically helps getting tweets from the specified location.
2. Everytime, the tweets are collected and stored in dataframes. Now, all the dataframes are combined into one to hold all the data together. For this, we used the '`bind_rows()`' function.
3. From the collected data, we select only the necessary fields using the '`select()`' function.
4. Now, we write the collected data into a CSV file for storage.

## 2.3 COMMON CRAWL

Common Crawl refers to the organization that provides massive sets of data collected from the internet to the public for free; which is collected by crawling. It involves crawling through the required domains in this case 'CRIME' from [www.crimedaily.com](http://www.crimedaily.com) , [www.crimereports.com](http://www.crimereports.com) etc.. 'WGET' function helps download the 'war.gz' files from the domain. The next step involves extracting the required URLs from the WARC files. The processing was done in python. This can be done by mentioning the topic and sub-topics as keywords. We collected a total of 599 URLs for 'CRIME'.

## 3. DATA PREPROCESSING/ DATA CLEANING

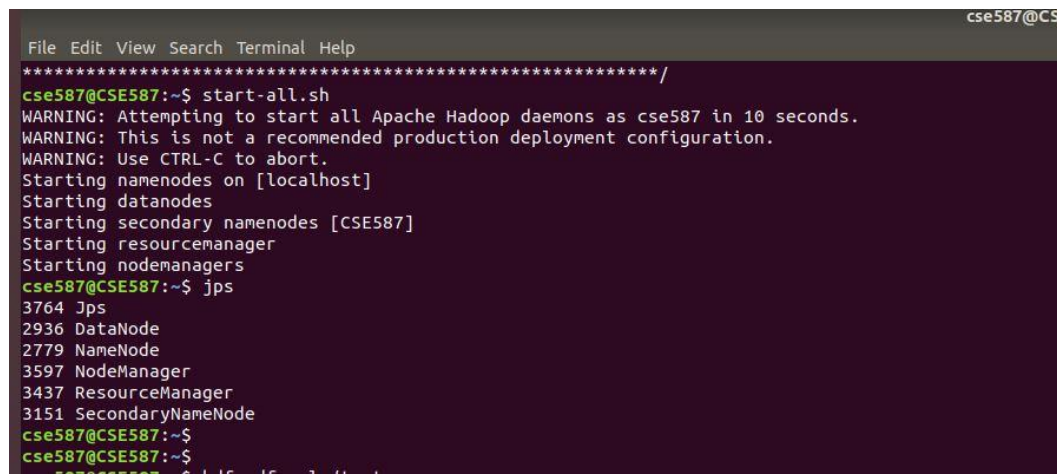
Data pre-processing is used since real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. The data collected from NYTimes, Twitter and Common Crawl need to be stripped clean before running the HDFS structure. The steps involved for data cleaning are:

1. The text extracted from URLs are stored in text files. In case of Twitter, the text field is extracted from the CSV file and stored separately.
2. This text file is subjected to a series of processing. Initially, we tokenize the text i.e. the text is broken down into single words.
3. Now, each word is tested for the following:
  - Check if it alphabetic using `isalpha()` function. We strip out all numeric terms.
  - Remove punctuations.
  - Replace multiple white-spaces by a single white-space.
  - Remove the word if it belongs to the list of stopwords.
  - Remove the words that have a high word count because of good usage in conversations. Eg: "said", "could", "would" etc.
4. In case of twitter data, it requires some additional data processing like:
  - Remove the links that are a part of the tweet text field starting with `http` or `https`.
  - Remove emoticons.
  - Remove screen names if present i.e. the names starting with '@'.
5. In case of common crawl data, we use Beautiful Soup package to help with html parsing.

6. Finally, once the clean text is secured, stemming/lemmatization is applied. This concluding data is stored and used as an input to the MapReduce structure.

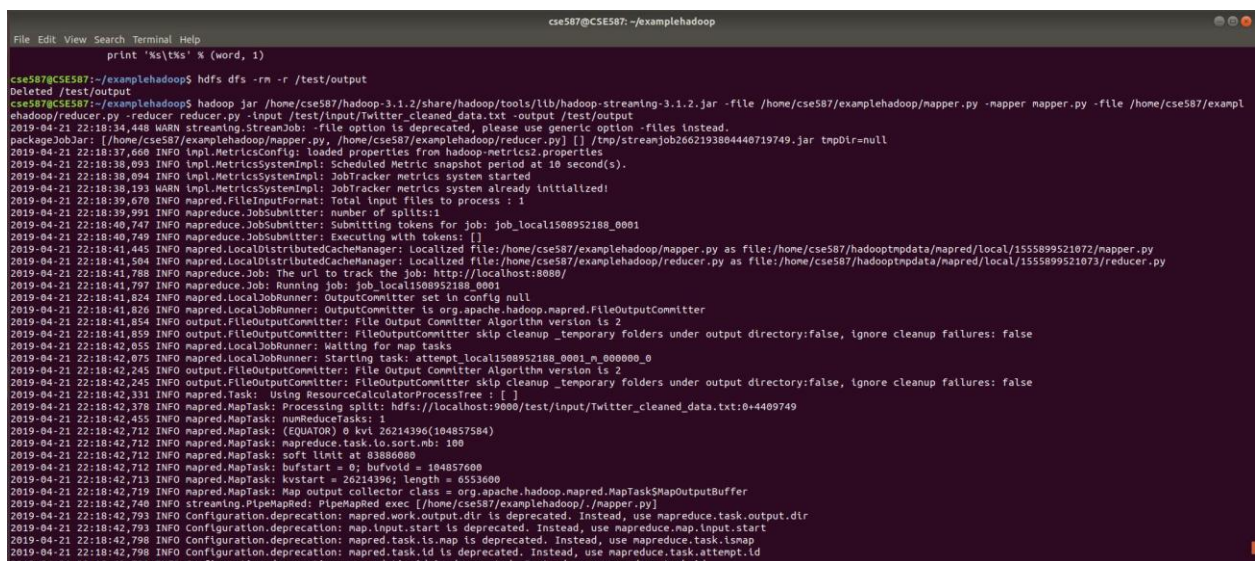
## 4. WORD COUNT

Word count is computed using a MapReduce Structure. This is done with the help of the mapper.py and reducer.py scripts given. The final cleaned text is given as input to the mapper. The mapper produces (key, value) pairs as output i.e. it emits <word, 1> list. The reducer gets input from the shuffling phase. It aggregates the list of values for each key and finally produces the list of co-occurring words with sums. From this, we retrieve the top 10 most frequently used words in the document. This is done for the texts from NYTimes, Twitter and Common Crawl.



```
File Edit View Search Terminal Help
*****/
cse587@cSE587:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as cse587 in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [CSE587]
Starting resourcemanager
Starting nodemanagers
cse587@cSE587:~$ jps
3764 Jps
2936 DataNode
2779 NameNode
3597 NodeManager
3437 ResourceManager
3151 SecondaryNameNode
cse587@cSE587:~$
cse587@cSE587:~$
```

Fig 1. Starting namenode and datanode on hdfs



```
File Edit View Search Terminal Help
print '%s\t%s' % (word, 1)
cse587@cSE587:~/examplehadoop$ hdfs dfs -rm -r /test/output
Deleted /test/output
cse587@cSE587:~/examplehadoop$ hadoop jar /home/cse587/hadoop-3.1.2/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -file /home/cse587/examplehadoop/mapper.py -mapper mapper.py -file /home/cse587/examplehadoop/reducer.py -reducer reducer.py -input /test/input/Twitter_cleaned_data.txt -output /test/output
2019-04-21 22:18:34,448 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
2019-04-21 22:18:37,660 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-21 22:18:38,093 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-21 22:18:38,094 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-21 22:18:39,193 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-21 22:18:39,670 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-21 22:18:39,991 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-21 22:18:40,747 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1508952188_0001
2019-04-21 22:18:40,749 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-21 22:18:41,445 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/examplehadoop/mapper.py as file:/home/cse587/hadooptmpdata/mapred/local/1555899521072/mapper.py
2019-04-21 22:18:41,504 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/examplehadoop/reducer.py as file:/home/cse587/hadooptmpdata/mapred/local/1555899521073/reducer.py
2019-04-21 22:18:41,788 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-21 22:18:41,797 INFO mapreduce.Job: Running job: job_local1508952188_0001
2019-04-21 22:18:41,824 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2019-04-21 22:18:41,826 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2019-04-21 22:18:41,854 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2019-04-21 22:18:41,859 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 22:18:42,055 INFO mapred.LocalJobRunner: Waiting for map tasks
2019-04-21 22:18:42,075 INFO mapred.LocalJobRunner: Starting task: attempt local1508952188_0001_n_000000_0
2019-04-21 22:18:42,245 INFO output.FileOutputCommitter: FileOutputCommitter Algorithm version is 2
2019-04-21 22:18:42,245 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 22:18:42,331 INFO mapred.MapTask: Using ResourceCalculatorProcessTree : [ ]
2019-04-21 22:18:42,378 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/test/input/Twitter_cleaned_data.txt:0:4489749
2019-04-21 22:18:42,455 INFO mapred.MapTask: numReduceTasks: 1
2019-04-21 22:18:42,712 INFO mapred.MapTask: (EQUATOR) 0 kvt, 26214396(104857584)
2019-04-21 22:18:42,719 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2019-04-21 22:18:42,712 INFO mapred.MapTask: soft limit at 83886080
2019-04-21 22:18:42,712 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2019-04-21 22:18:42,713 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2019-04-21 22:18:42,719 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2019-04-21 22:18:42,793 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
2019-04-21 22:18:42,793 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
2019-04-21 22:18:42,798 INFO Configuration.deprecation: mapreduce.task.is.map is deprecated. Instead, use mapreduce.task.ismap
2019-04-21 22:18:42,798 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
2019-04-21 22:18:42,799 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.id
```

Fig 2. Executing word count for Twitter data

Word	Count
welcome	57
welcomed	6
welcometogilead	1
welcomfrom	1
welcoming	4
welder	2
welding	4
welfare	56
welfarereformnow	1
welkm	1
well	654
wellbeing	2
wellandsfaith	1
weller	1
welles	1
wellington	2
wellis	1
wellrape	1
wellroundednut	1
wellsfargo	10
wellzzach	1
welocne	1
welp	1
welshalan	6
wembley	3
wemurdertrees	2
wen	6
wench	1
wende	1
wendy	4
wendydavis	2
wendyrsherman	1
wendys	3
wendystegelman	1
wendywatching	1
wendywilliams	5
wendyyytraka	1
weneedalaw	1
weneedarealpresident	1
weneediversebooks	1
went	263
wentworth	2
wenyka	1

**Fig 3.** Word Count on Twitter Data

## 5. WORD CO-OCCURRENCE

Word co-occurrence involves the phenomenon of computing the frequency of highly co-occurring words in the given document. This is also done using the mapper and reducer scripts which are written in python. The reducer gets input from the shuffling phase. It aggregates the list of values for each key and finally produces the list of co-occurring words with sums. From this, we retrieve the top 10 most frequently appearing co-occurring words. This is done for the text from NYTimes, Twitter and Common Crawl.

```

File Edit View Search Terminal Help
cse587@CSE587: ~/examplehadoop
cse587@CSE587:~/examplehadoop$ hadoop jar /home/cse587/hadoop-3.1.2/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -file /home/cse587/examplehadoop/mapper-co.py -mapper mapper-co.py -file /home/cse587/examplehadoop/reducer.py -reducer reducer.py -input /test/input/Twitter_cleaned_data.txt -output /test/output
2019-04-21 22:43:34,753 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/cse587/examplehadoop/mapper-co.py, /home/cse587/examplehadoop/reducer.py] [] /tmp/streamjob4274381204334250984.jar tmpDir=null
2019-04-21 22:43:37,541 INFO Impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-21 22:43:37,847 INFO Impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-21 22:43:37,848 INFO Impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-21 22:43:37,966 WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-21 22:43:39,315 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-21 22:43:39,594 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-21 22:43:40,372 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local212077994_0001
2019-04-21 22:43:40,373 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-21 22:43:41,083 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/examplehadoop/mapper-co.py as file:/home/cse587/hadooptmpdata/mapred/local/1555901020700/mapper-co.py
2019-04-21 22:43:41,165 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/examplehadoop/reducer.py as file:/home/cse587/hadooptmpdata/mapred/local/1555901020701/reducer.py
2019-04-21 22:43:41,487 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-21 22:43:41,495 INFO mapreduce.Job: Running job: job_local212077994_0001
2019-04-21 22:43:41,515 INFO mapred.LocalJobRunner: OutputCommitter set to config null
2019-04-21 22:43:41,517 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2019-04-21 22:43:41,546 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2019-04-21 22:43:41,546 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 22:43:41,775 INFO mapred.LocalJobRunner: Waiting for map tasks
2019-04-21 22:43:41,788 INFO mapred.LocalJobRunner: Starting task: attempt local212077994_0001_m_000000_0
2019-04-21 22:43:41,935 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2019-04-21 22:43:41,940 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 22:43:42,059 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2019-04-21 22:43:42,101 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/test/input/Twitter_cleaned_data.txt:0+4409749
2019-04-21 22:43:42,203 INFO mapred.MapTask: numReduceTasks: 1

```

**Fig 4.** Executing Word Co-occurrence on twitter data

```
cse587@CSE587: ~/examplehadoop
File Edit View Search Terminal Help
Reduce input records=490531
Reduce output records=313831
Spilled Records=981062
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=176
Total committed heap usage (bytes)=457318400
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=4409749
File Output Format Counters
Bytes Written=5313975
2019-04-21 22:43:57,687 INFO streaming.StreamJob: Output directory: /test/output
cse587@CSE587:~/examplehadoop$ hdfs dfs -cat /test/output | head -20
cat: '/test/output': Is a directory
cse587@CSE587:~/examplehadoop$ hdfs dfs -cat /test/output/part-00000 | head -20
aa benefited 1
aa burning 1
aa charged 1
aa church 1
aa civil 1
aa controlled 1
aa enslavement 1
aa federal 1
aa flight 1
aa gay 1
aa knew 1
aa ozzie 1
aa passenger 1
aa pretending 1
aa room 1
```

**Fig 5.** Example Word Co-occurrence count

## 7. VISUALIZATION

Visualization is an essential part of this project as we are required to develop word clouds. Word Clouds are visual representation of text data used to portray the highlighted words i.e. the words with the highest frequencies. We use Tableau to perform this. We know that the output from the reducer emits the words list with their frequencies. This is imported into a CSV file and sorted in a non-increasing order based on the sum value. From this, it is easy to extract the top 10 words appearing in Twitter data, NYTimes articles and Common Crawl.

An excel sheet with the top 10 list is imported into Tableau and is used to create the word clouds respectively. Word Clouds are also created for the top 10 list of co-occurring words from the respective sources.

## 8. CONCLUSION

Thus Word-Count and Word Co-occurrence problems have been implemented using MapReduce. This project has helped expand knowledge on data aggregation and visualization. It enabled working with different search APIs for Twitter and NYTimes and also working on the returned results. Thus this lab has aided with working on Big Data and analytical tools.

**Website link:** <https://public.tableau.com/profile/monisha5735#!/vizhome/Monisha/Sheet5>



## REFERENCES

1. <https://pypi.org/project/wget/>
2. <https://marcobonzanini.com/2015/03/23/mining-twitter-data-with-python-part-4-rugby-and-term-co-occurrences/>
3. <https://developer.rhino3d.com/guides/rhinopython/python-csv-file/>
4. <https://www.machinelearningplus.com/nlp/lemmatization-examples-python/>
5. <https://pythonspot.com/tokenizing-words-and-sentences-with-nltk/>