**Step-wise execution of Lab2 assignment**

**a.** **Query** : H1B

**Tweets collected using TwitterAPI** : 2000

**Articles collected using NYTimes API** : 50

**Python files attached for collecting the data** : tweet\_collect.py

**b. Hadoop used** : given VM image, tested basic word count example provided.

**c.** Loaded the data TwitterData and NewsData using ssh

**Steps:**

1.Added Network Adapter for allowing remote connection

2.Set static ip as 192.168.56.10 for Hadoop VM machine

3.Copied files using scp command through ssh

**Commands used :**

*scp /Users/sunitapattanayak/Documents/DIC/Lab2/NewsData/\* hadoop@192.168.56.10:/home/hadoop/NewsData*

*scp /Users/sunitapattanayak/Documents/DIC/Lab2/TwitterData/\* hadoop@192.168.56.10:/home/hadoop/TwitterData*

Folder Structure as shared :

/home/hadoop/TwitterData/resu1.txt, resu1\_2.txt : each contains 1000 tweets total:2000

/home/Hadoop/NewsData/resu4.txt, resu4\_0.txt, resu4\_1.txt, resu4\_3.txt, resu4\_6.txt, resu4\_7.txt, resu4\_8.txt : total articles : 50

**d.** **Language Used** : python

**Mapper.py** : Clean data, removes stop words by using nltk

**Reducer.py** : Counts the useful words from output of Mapper.py

**Steps:**

1.Create Mapper.py that reads each page or text files, cleans them and outputs each word with count 1 as output.

2.Create Reducer.py that count the no. of occurrences of each word and outputs each word with it frequency.

3.Put the output of previous step I.e twitter words text file and NY times words text file in hfs by using following commands:

*hdfs dfs -put NewsData/\* lab2*

*hdfs dfs -put TwitterData/\* lab2*

4.Run jar file available to run map reduce function by using following command :

*hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -mapper mapper.py -reducer reducer.py -input lab2 -output lab2\_output*

*hdfs dfs –cat lab2\_output/\* >> output.txt*

**e,f,g.** Use output.txt file to visualize using d3.js and see it interactively on the webpage.

**Files used** : D3.layout.cloud.js,trial.htm,script.js

**Intially data collected** : 1000+20

**Final data collected** : 2000+50

**h.** Created modified mapper reducer files to search top ten words and find their cooccurrence in each paragraph of articles and each tweets.

1.**Mapper1.py** : Clean data, removes stop words by using nltk , find occurrence of each top ten word pairs and output tab separated words with 1 as their count

2.**Reducer.py** : count the no. of occurrences of each word pair and outputs tab separated word pair with its frequency.

3.Put the output of previous step I.e twitter words text file and NY times words text file in hfs by using following commands:

*hdfs dfs -put NewsData/\* lab2\_ext*

*hdfs dfs -put TwitterData/\* lab2\_ext*

4.Run jar file available to run map reduce function by using following command :

*hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -mapper mapper1.py -reducer reducer.py -input lab2\_ext -output lab2\_ext\_output*

*hdfs dfs –cat lab2\_output/\* >> output\_ext.txt*

**Created convert.py** : It converts the text file created into csv file(output\_ext.txt to output\_ext.csv) which can be easily read by the d3.js code for an interactive website designing.

**Steps for running other data :**

1.Run the following commands:

./delete.sh

./hidden.sh

CSV files created :

out.txt and out.csv : WordCloud of words

out\_ext.txt and out\_ext.csv : WordCloud of pairwise top 10 words

P.S.: Please rename text file containing articles to be run as file.txt or change the name of file in hidden.sh

2.Run the following d3.js files:

wordcloud\_input.html: Input text in text area to save as file along with filename, preferably file.txt which can be downloaded in Downloads folder.

wordcloud.html: For wordcloud of top 50 occurring words in the articles

cooccurencewordcloud.html: For wordcloud of pairwise occurring top 10 words in articles

**Elaborate steps inside the shell script files :**

1. Please place the text file in another folder, say “TestData”
2. *hdfs dfs -put TestData/\* lab2\_test*
3. *hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -mapper mapper.py -reducer reducer.py -input lab2\_test -output lab2\_test\_output*
4. *hdfs dfs –cat lab2\_test\_output/\* >> output\_test.txt*
5. Please change text file name to output\_test.txt and csv file name to output\_test.csv in convert.py to obtain csv file for interactive wordcloud output
6. Please change the input file name in script.js to output\_test.csv
7. *hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -mapper mapper1.py -reducer reducer.py -input lab2 -output lab\_test\_ext\_output*
8. *hdfs dfs –cat lab\_test\_ext\_output/\* >> output\_test\_ext.txt*
9. Please change text file name to output\_test\_ext.txt and csv file name to output\_test\_ext.csv in convert.py to obtain csv file for interactive wordcloud output
10. Please change the input file name in script.js to output\_test\_ext.csv

The hdfs directory structure should contain the following :

lab2\_test

lab2\_test\_output

lab2\_test\_ext\_output

The Hadoop home directory structure should contain the following :

output\_test.txt and output test.csv : WordCloud of words

output\_test\_ext.txt and output\_text\_ext.csv : WordCloud of pairwise top 10 words

Other commands used in the project:

hdfs dfs –ls : To list all contents in hdfs

hdfs dfs –rm –r foldername: To delete folders from hdfs

**Explaining the html and javascript files:**

Wordcloud.html: Creates a wordcloud for first 50 highest occurring words

cooccurencewordcloud.html: Creates a wordcloud for pairwise occurrence of top 10 words from first output file

wordcloud\_input.html: Text Area to save as file along with filename, preferably file.txt

script1.js, script2.js : Java Script for creating wordclouds

wordcloud java scripts files script.js and d3.layout.cloud.js : Wordcloud layout by Jason Davies, <http://www.jasondavies.com/word-cloud/> and modified as per requirements of the project.



