

# **DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION**

**Kishan Damotharan (kishandh)**

**Sriram Venkataramanan (sv84)**

## OBJECTIVE :

- (i) Data aggregation from more than one source using the APIs (Application programming interface) exposed by data sources.
- (ii) Applying classical big data analytic method of MapReduce to the unstructured data collected.
- (iii) Store the data collected on WORM infrastructure Hadoop.
- (iv) Building a visualization data product.

## TOPIC OF THE PROJECT: BASKETBALL

We decided to collect data related to Basketball. **Basketball is played both at the pro level and college level.** The pro level basketball is termed as **NBA** and the college level basketball is termed as **NCAA**. Thus our subtopics related to Basketball are:

- NBA
- NCAA

The above mentioned objectives are developed and the process is documented below.

### OBJECTIVE 1: Data aggregation from API

#### Requirement:

We should aim to collect at least 500 articles for each topic (so maybe 100 articles for each sub-topic) from NY Times, at least 500 articles from Common Crawl and at least 20000 tweets to perform analysis. These 20000 tweets should be unique tweets i.e. all duplicate tweets and retweets should be removed.

#### Development process:

1) According to the requirement mentioned above, we collected data from 3 sources.

#### I. New York Times Articles

- 1) Data from New York Times has been obtained by writing a script in Python. We have included the file in our submission and the name of the file is **nytimes.py**.
- 2) We collected articles for every sub topics.
- 3) We removed **stop words** and performed **stemming** using **nlTK library**.
- 4) While collecting data from Ny Times, the API would return 10 url's per page and we used **beautiful soup library to scrape the data from these Url's**
- 5) The number of articles is 430 in NBA and 200 in NCAA. The total number of articles collected is 630.

#### II. Twitter:

- 1) Data from twitter has been obtained by writing a program in R using twitterR package. We have included the file in our submission and the name of the file is **tweet.ipynb**
- 2) We collected tweets by using key words for every subtopic. NBA related tweets has been Obtained by using the key word "NBA""#NBA" and NCAA related tweets has been obtained by Using the key word "NCAA""#NCAA".
- 3) Once the data is collected, in-order to perform pre-processing we used a separate python script processTweets.py
- 4) We removed stop words and performed stemming using **nlTK library**.

- 5) The **number of tweets collected is 36000 total together.**

### III. Common Crawl

- 1) Data from Common crawl has been obtained by writing a script in Python. We have included the file in our submission and the name of the file is commonCrawlerNba.ipynb and commonCrawlerNcaa.ipynb and commonCrawlerNcaa2.ipynb.
- 2) We collected articles for every subtopic. In-order to collect data we used specific domain and index to extract data. We formed the Url and the API returns an WARC, Header and response for the url.
- 3) We processed those responses into a **beautiful soup** object and performed data extraction process.
- 4) The **number of articles is totally 2000 altogether.**


### OBJECTIVE 2: Applying classical big data analytic method of MapReduce to the unstructured data collected

#### AWS:




We used AWS in order to run the Map Reduce Code using Hadoop.

#### STEPS FOLLOWED:





- 1) S3 storage: Initially we created a bucket in S3 storage named as dicproject2

<input type="checkbox"/> Bucket name ▾	Access ⓘ ▾	Region ▾	Date created ▾
<input type="checkbox"/>  dicproject2	Bucket and objects not public	US East (N. Virginia)	Apr 16, 2019 9:06:22 PM GMT-0400

- 2) Then we created a folder structure in the following way in order to upload the data we have collected from three sources so far.

<input type="checkbox"/>  CC	--	--	--
<input type="checkbox"/>  NYC	--	--	--
<input type="checkbox"/>  TWITTER	--	--	--

- 3) Inside each folder we uploaded the data we have collected for our two subtopics: NBA and NCA.






<input type="checkbox"/>  nba_input	--	--	--
<input type="checkbox"/>  nba_output	--	--	--
<input type="checkbox"/>  nba_woc_output	--	--	--
<input type="checkbox"/>  ncaa_input	--	--	--
<input type="checkbox"/>  ncaa_output	--	--	--
<input type="checkbox"/>  ncaa_woc_output	--	--	--

- 4) The nba\_input and ncaa\_input consists of the data related to respective subtopics, the source in this case is New York Times. The folder structure is similar to other sources as well.
- 5) Once the data have been uploaded, we have to run the mapper and reducer code in order to obtain the outputs.

### CASE 1: Word Count

The word count Mapper and Reducer code is written in two python files : mapper.py and reducer.py and both the codes are uploaded in the same path where the CC, NYC and Twitter folder present in the above picture.

Word count reducer simply returns a word along with number of times the word has appeared in the whole data. We used a separate python code to get the top 10 words of highest count

<input type="checkbox"/>		CC	--	--	--
<input type="checkbox"/>		NYC	--	--	--
<input type="checkbox"/>		TWITTER	--	--	--
<input type="checkbox"/>		mapper.py	Apr 16, 2019 9:31:51 PM GMT-0400	561.0 B	Standard
<input type="checkbox"/>		reducer.py	Apr 16, 2019 9:32:05 PM GMT-0400	1.0 KB	Standard

### CASE 2: WORD OCCURRENCE:

The word count Mapper and Reducer code is written in two python files : mapper.py and reducer.py and both the codes are uploaded in the same path where the CC, NYC and Twitter folder present in the above picture.

Word co-occurrence is nothing but finding the highest co-occurred word from the top words found in word count.

<input type="checkbox"/>		mapper_wordocc.py	Apr 18, 2019 1:16:54 AM GMT-0400	769.0 B	Standard
<input type="checkbox"/>		reducer_wordocc.py	Apr 18, 2019 1:16:55 AM GMT-0400	1.0 KB	Standard

6) The next step is to run the mapper and reducer code in Hadoop by creating a cluster in EMR.

**Steps in AWS to run the map reduce code once the Data structure has been created.**

- Go to Services → EMR → Create Cluster → Advanced options
- Select Hadoop 2.8.5 from the Software Configuration section

#### Software Configuration

Release	emr-5.23.0	
<input checked="" type="checkbox"/>	Hadoop 2.8.5	<input type="checkbox"/> Zeppelin 0.8.1
<input type="checkbox"/>	JupyterHub 0.9.4	<input type="checkbox"/> Tez 0.9.1

- Then Choose Streaming program and give the location of mapper, reducer code and the location of input and give the location where you want to obtain the output by clicking the configure option.

## Add steps (optional) ?

Step type Streaming program

[Configure](#)

☒ Auto-terminate cluster after the last step is completed

Step type Streaming program

Name\* Streaming program

Mapper\*

s3://

S3 location of the map funct

Reducer\*

s3://

S3 location of the reduce fui  
Hadoop streaming comman

Input S3 location\*

s3://

s3://<bucket-name>/<folder>/

Output S3 location\*

s3://

s3://<bucket-name>/<folder>/

Arguments

Action on failure Continue

What to do if the step fails.

- Once you have given the above mentioned locations, you can proceed to further steps and create the Cluster. If necessary we can log the process.
- The cluster initially would be in Running State, once it gets terminated without errors, outputs of the code would be obtained in the folders we mentioned.

7) The step 6 is same for both the word count and word occurrence process.

8) The outputs obtained for one case is shown below:

<input type="checkbox"/>	_SUCCESS	Apr 16, 2019 9:48:09 PM GMT-0400	0 B	Standard
<input type="checkbox"/>	part-00000	Apr 16, 2019 9:48:00 PM GMT-0400	16.6 KB	Standard
<input type="checkbox"/>	part-00001	Apr 16, 2019 9:48:00 PM GMT-0400	17.2 KB	Standard
<input type="checkbox"/>	part-00002	Apr 16, 2019 9:48:00 PM GMT-0400	16.8 KB	Standard
<input type="checkbox"/>	part-00003	Apr 16, 2019 9:48:07 PM GMT-0400	16.7 KB	Standard
<input type="checkbox"/>	part-00004	Apr 16, 2019 9:48:06 PM GMT-0400	16.2 KB	Standard
<input type="checkbox"/>	part-00005	Apr 16, 2019 9:48:08 PM GMT-0400	17.1 KB	Standard
<input type="checkbox"/>	part-00006	Apr 16, 2019 9:48:09 PM GMT-0400	16.4 KB	Standard

Activate Win

9) This indicates the word count was obtained in 7 reducers.

10) The files can be downloaded for data visualisation purpose.

### TYPES OF MAP REDUCE RUN:

- We ran the map reduce code individually for every sub topic in our data source i.e. for NBA and NCAA present in 3 different sources: 1) NYT 2) Common crawl 3) Twitter. **Totally we ran the MR code 6 times.**
- We then ran the map reduce code once for every topic i.e. for overall data from sources: 1) NYT 2) Common crawl 3) Twitter. **Totally we ran the MR code 3 times.**
- We ran the map reduce code **once on the overall data** from all three sources clubbed together.
- All these different runs are useful for visualizing the data below.

### OBJECTIVE 3: Data Visualisation

For data visualisation we used HTML, Javascript and used D3.js for visualisation purposes.

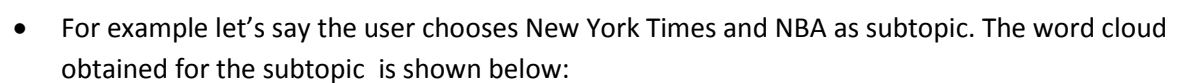
[illegible]

An image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.

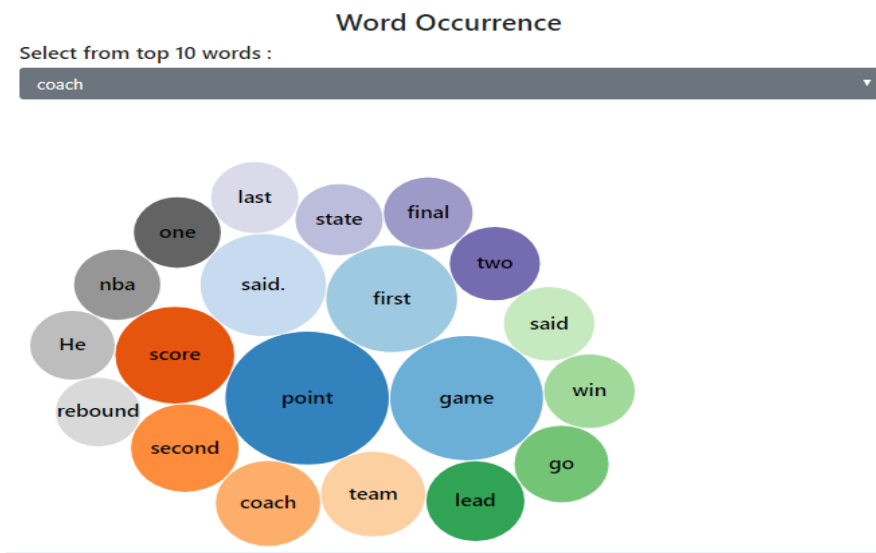
**Bigram:** Used for visualising word co-occurrence.

When a user chooses a word from a topic from the UI, it displays the top words that co-occurred along with that particular word in the form of a bigram. The size of the bubble in the bigram indicates the frequency of the words.

- The right side of the UI shows the word count of a particular subtopic chosen in the Source that is displayed above.
- The user is given the option of selecting the source from a drop down menu as below:



- The above figure shows the top words that is obtained for the subtopic NBA in NYT in the form of a word cloud.
- We can see from the image above, **the word point has occurred the most number of times** since it is shown the **biggest among the all**.
- For the same example i.e NYT and Nba as subtopic, user chooses a word from the Select Top 10 words option. The bigram i.e. the word co-occurrence for the selected word is shown below

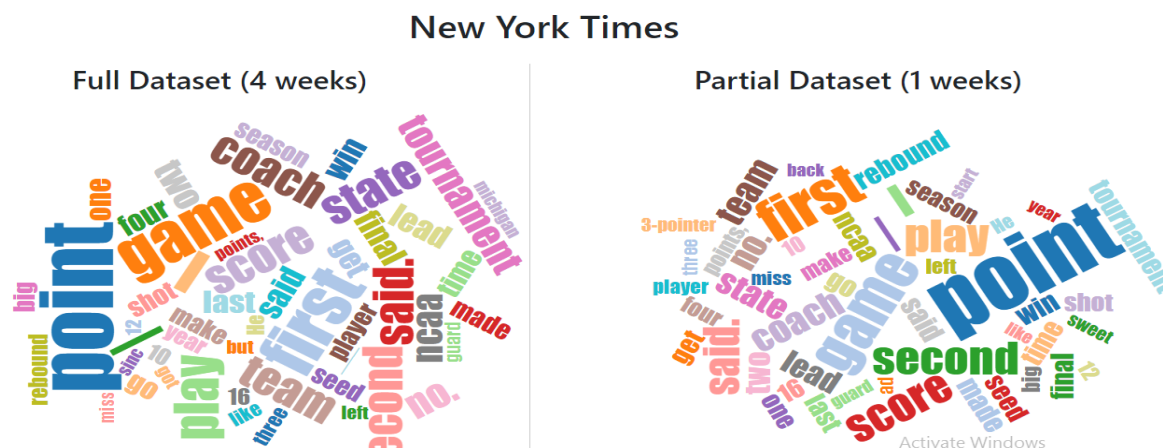


- Thus these are the Co-occurred words for the chosen word coach.
- Thus the word count and word co-occurrence for the data we collected have been properly visualized and built as web app.

### Additional visualisation

- The user can also view how the word cloud varies for a smaller data set(i.e 1 week) and bigger data set(i.e 4 weeks).
- The comparison is obtained for all the three datasets.1)New York Time 2) Commoncrawl 3) Twitter

### New York Time comparison



## Common Crawl

### Full Dataset (4 weeks)



### Partial Dataset (1 weeks)



## Twitter comparison

Full Dataset (4 weeks)



Twitter

### Partial Dataset (1 weeks)



**Full Data Cloud** : All sources clubbed together.

## Full dataset

## Word Count



Thus we can see from the above pics, how nicely **the word converges from a smaller data set to a bigger data set**. The words remain almost similar and size gets increases as we move on to a bigger data set from smaller one.



## CONCLUSION:

- All the development process is documented properly.
- We learnt how to do data aggregation and data collection from multiple sources, say New York Times, Twitter and Common crawl using the API provided by python/R
- We learnt the concept of map reduce concept and how to implement it in an AWS infrastructure. The map reduce code were implemented in python
- Data visualisation is also done using Html, Css, Javascript and d3.js library.
- The main learning of this project is to perform data analysis by developing an interactive web design app.