DATA INTENSIVE COMPUTING: LAB 2

DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

TEAM MEMBERS:
EMMANUEL JOHNSON GNANA CHANDRA JOHNSON,
VENKTESH KAVIARASAN

PERSON NO: 50290792, 50289400

ABSTRACT:

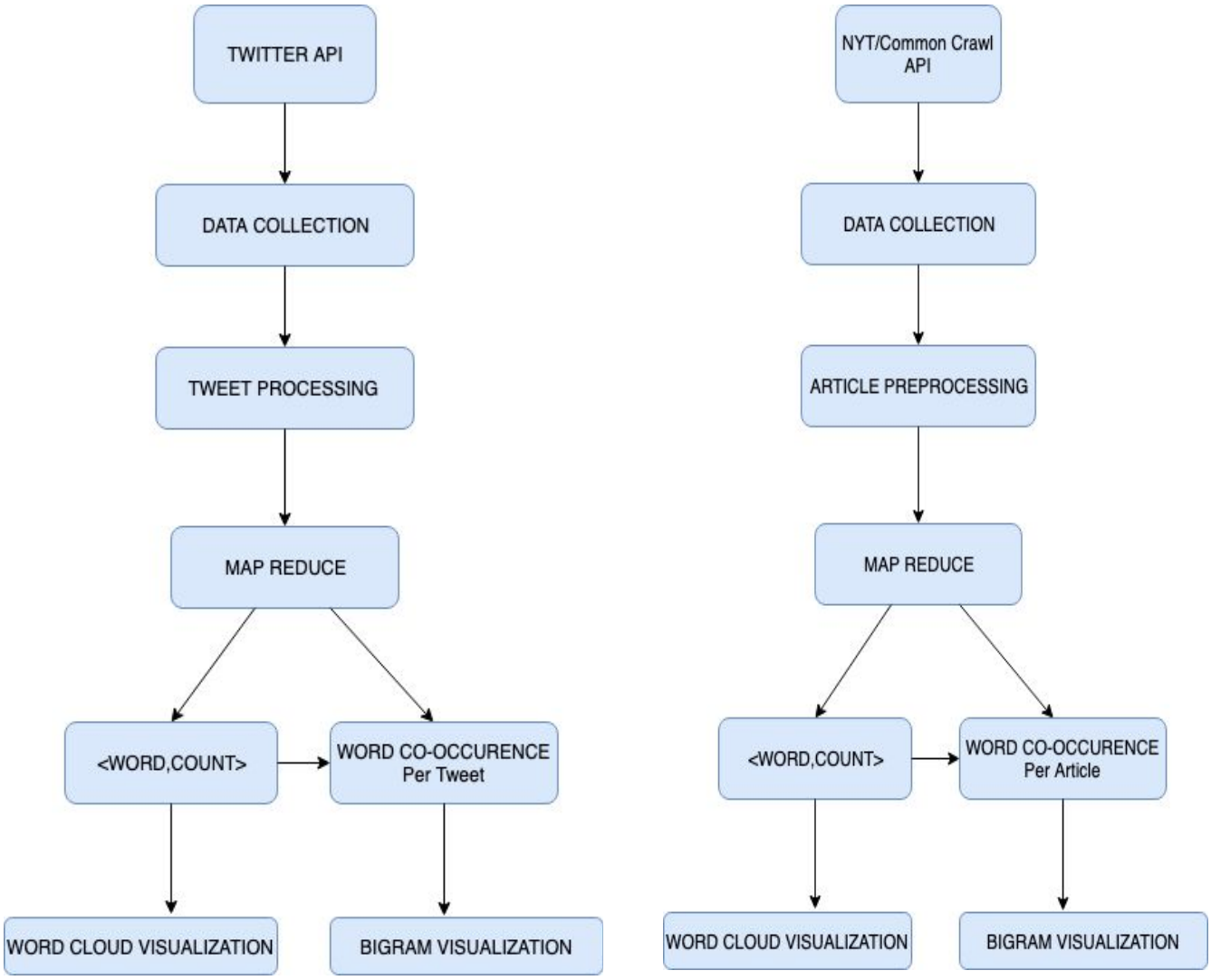
The objective of this project is to collect data from three different sources such as twitter, new york times and common crawl using their apis. Then setup a hadoop platform and run mapreduce for the unstructured data collected. We do analysis like word count and word co-occurrence using the hadoop mapreduce on the data collected. We then visualize the output using d3.js.

DATA COLLECTION AND PROCESSING:

- The topic chosen for data collection is "SPORTS" wherein the subtopics which were chosen are "BASKETBALL" , "FOOTBALL", "BASEBALL" and "GOLF".
- ***Tweet Collection:*** Tweets were collected using Tweepy package in python from each of the above mentioned subtopics. Retweets and duplicate tweets were removed while collecting in order to make the data clean for processing.
- ***NY Times Articles:*** In NY Times appropriate URL's were formed by using the query term depending on the sport name and then they were crawled, the resulting HTML responses were parsed by using Beautiful Soup 4 framework in python to obtain just the article body, thereby the resulting cleaned articles were saved in separate files for processing.
- ***Common Crawl Articles:*** From the Common Crawl API appropriate indexes were chosen for crawling and the domain chosen was : USA TODAY for collecting sports related news. Using the generated URL's the common crawl API was hit and the resulting JSON response contained paths to compressed files in Amazon S3 Bucket. These (.WARC) files were then unzipped out and parsed using Beautiful Soup framework in python.
- ***Data Processing:*** We have used nltk libraries to remove the stop words and stem the words. We have also remove punctuations, digits, unwanted symbols, emoticons and urls from the articles and tweets using regex. Thereby the data was cleaned and is ready to be fed into the hadoop file system.

DATA SOURCE	DATA COLLECTED
TWITTER	~ 46,000 Tweets
NEW YORK TIMES	~1400 Articles
COMMON CRAWL	~ 600 Articles

BIG DATA HADOOP PIPELINE:



HADOOP AND MAP-REDUCE:

- We have setup our hadoop environment using docker and the data in the local file system is mounted with the hadoop docker container.
- We create a directory in our hadoop file system and copy all the crawled data into the hadoop file system folder that we have created. The below image shows how we've put the twitter articles into the hadoop file system.

```
[root@quickstart data]# hadoop fs -mkdir /user/dic/MR/tw_inputs
[root@quickstart data]# hadoop fs -put tw/*.txt /user/dic/MR/tw_inputs
```

- Now we run the mapreduce for word count for all the files. We've run the mapreduce for word count and word co-occurrence in the following combinations,
    - All the subtopics for Twitter
    - All the subtopics for New York Times
    - All the subtopics for Common Crawl
    - Each subtopics (4) for Twitter
    - Each subtopics (4) for New York Times
    - Each subtopics (4) for Common Crawl
- Below is the screenshot of how we execute the mapreduce for all subtopics in twitter data. The first image is mapreduce for word count and the following image is mapreduce for word co-occurrence.

```
[root@quickstart data]# hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.7.0.jar -file /src/part3/twitter/code/m
apper.py -mapper /src/part3/twitter/code/mapper.py -file /src/part3/twitter/code/reducer.py -reducer /src/part3/twitter/code/red
ucer.py -input /user/dic/MR/tw_input/tw*.txt -output /user/dic/MR/tw_wc
19/04/22 01:59:30 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/src/part3/twitter/code/mapper.py, /src/part3/twitter/code/reducer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.
0.jar] /tmp/streamjob6483725864204256427.jar tmpDir=null
19/04/22 01:59:33 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/04/22 01:59:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/04/22 01:59:35 INFO mapred.FileInputFormat: Total input paths to process : 4
19/04/22 01:59:35 INFO mapreduce.JobSubmitter: number of splits:4
19/04/22 01:59:35 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1555283385854_0089
19/04/22 01:59:35 INFO impl.YarnClientImpl: Submitted application application_1555283385854_0089
19/04/22 01:59:36 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1555283385854_
0089/
19/04/22 01:59:36 INFO mapreduce.Job: Running job: job_1555283385854_0089
19/04/22 01:59:45 INFO mapreduce.Job: Job job_1555283385854_0089 running in uber mode : false
19/04/22 01:59:45 INFO mapreduce.Job:  map 0% reduce 0%
19/04/22 01:59:59 INFO mapreduce.Job:  map 50% reduce 0%
19/04/22 02:00:00 INFO mapreduce.Job:  map 100% reduce 0%
19/04/22 02:00:08 INFO mapreduce.Job:  map 100% reduce 100%
19/04/22 02:00:08 INFO mapreduce.Job: Job job_1555283385854_0089 completed successfully
19/04/22 02:00:08 INFO mapreduce.Job: Counters: 49
```

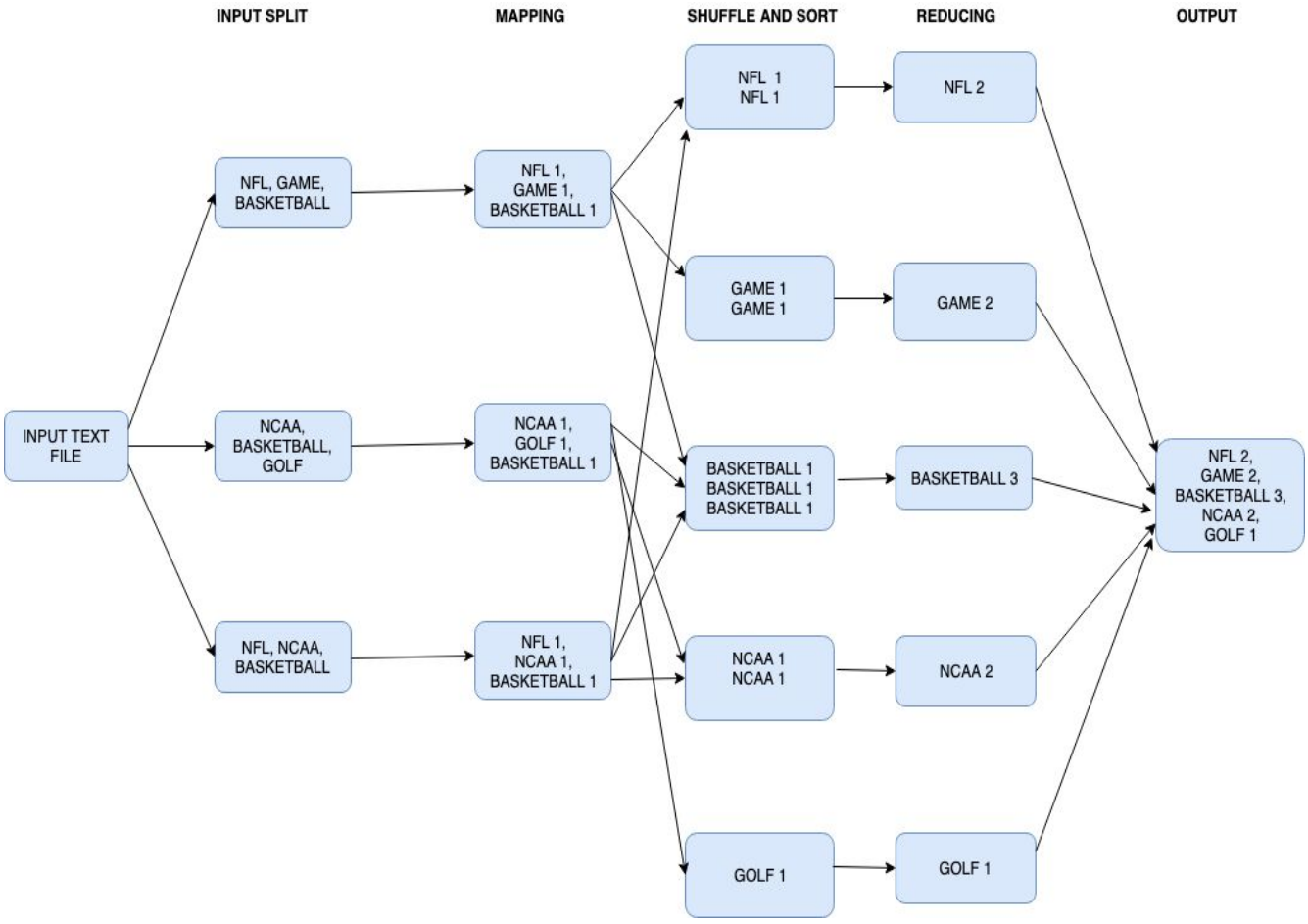
```
[root@quickstart data]# hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.7.0.jar -file /src/part3/twitter/code/m
apper_cowc_all.py -mapper /src/part3/twitter/code/mapper_cowc_all.py -file /src/part3/twitter/code/reducer_cowc.py -reducer /src
/part3/twitter/code/reducer_cowc.py -input /user/dic/MR/tw_input/tw*.txt -output /user/dic/MR/tw_wcoc
19/04/22 02:00:25 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/src/part3/twitter/code/mapper_cowc_all.py, /src/part3/twitter/code/reducer_cowc.py] [/usr/jars/hadoop-streaming
-2.6.0-cdh5.7.0.jar] /tmp/streamjob6160892689630521769.jar tmpDir=null
19/04/22 02:00:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/04/22 02:00:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/04/22 02:00:29 INFO mapred.FileInputFormat: Total input paths to process : 4
19/04/22 02:00:29 INFO mapreduce.JobSubmitter: number of splits:4
19/04/22 02:00:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1555283385854_0090
19/04/22 02:00:30 INFO impl.YarnClientImpl: Submitted application application_1555283385854_0090
19/04/22 02:00:30 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1555283385854_
0090/
19/04/22 02:00:30 INFO mapreduce.Job: Running job: job_1555283385854_0090
19/04/22 02:00:38 INFO mapreduce.Job: Job job_1555283385854_0090 running in uber mode : false
19/04/22 02:00:38 INFO mapreduce.Job:  map 0% reduce 0%
19/04/22 02:00:50 INFO mapreduce.Job:  map 25% reduce 0%
19/04/22 02:00:51 INFO mapreduce.Job:  map 75% reduce 0%
19/04/22 02:00:52 INFO mapreduce.Job:  map 100% reduce 0%
19/04/22 02:00:59 INFO mapreduce.Job:  map 100% reduce 100%
19/04/22 02:00:59 INFO mapreduce.Job: Job job_1555283385854_0090 completed successfully
19/04/22 02:01:00 INFO mapreduce.Job: Counters: 49
```

- The above mentioned combinations are ran for both the small data and the big data.
- The mapper of the word count emits each word with count as 1 and after the shuffling and sorting phase, the reducer aggregates each word's count and the total count for one particular word is obtained as a key value pair.

- The mapreduce output is the obtained by executing the below command,

```
[root@quickstart data]# hadoop fs -get /user/dic/MR/tw_wc /src/
```

- This output is then fed into a python script which fetches the top 10 words.
- The same steps are followed for the word co-occurrence. The difference is, we emit the count as 1 with a co-occuring word which is present in the same paragraph (nyt/cc article) or the same tweet (twitter), only if the word and the co-occuring word appears in the top 10 words that is produced by the word count mapreduce.
- The below diagram explains the working of the hadoop mapreduce of word count for a given input,



- A csv file is generated from the mapreduce output using python script. This file is then used by our webpage to show visualizations.

VISUALIZATION :

Word Clouds and Bigram bubble charts are generated for the Map-Reduce Output using d3.js.

Word Cloud:

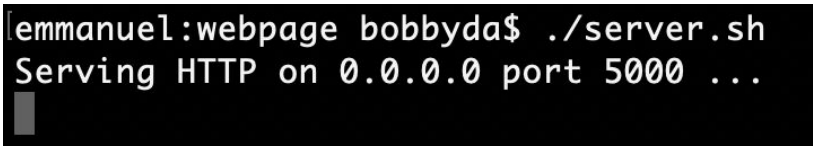
- d3.csv() function was used to read the data from the csv files. This function returns objects in the format of : {[text,size]} pairs. These objects are then passed to the d3.wordcloud() function to generate the corresponding word cloud. Thereby the word cloud was generated for the Top 10 words for each of data sources and their corresponding subtopics.
- In the word cloud, the size of the word is proportional to the number of times the word has appeared.

Word Co-Occurrence:

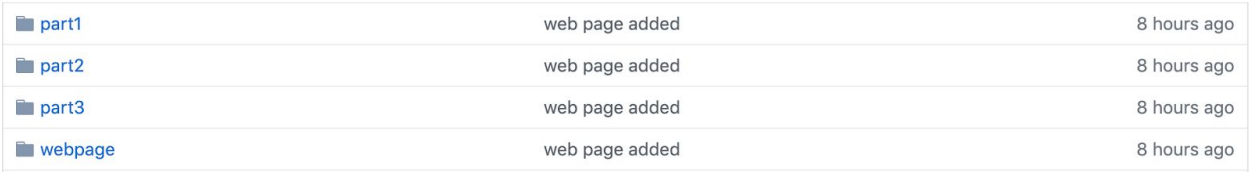
- We read the csv file using d3.csv() and format the data in {[Name,Count]} pairs. Then using this we create a bubble chart (bigram) using version 4 of d3.js.
- In the bigram, the size of the bubble is proportional to the number of times the word co-occurrence pair has appeared.

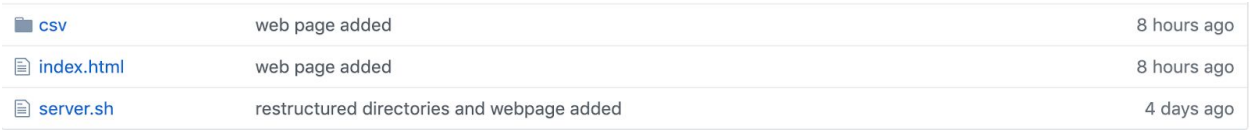
Running the web page:

- To start view the visualizations, we have to start our simple python server using the following command,



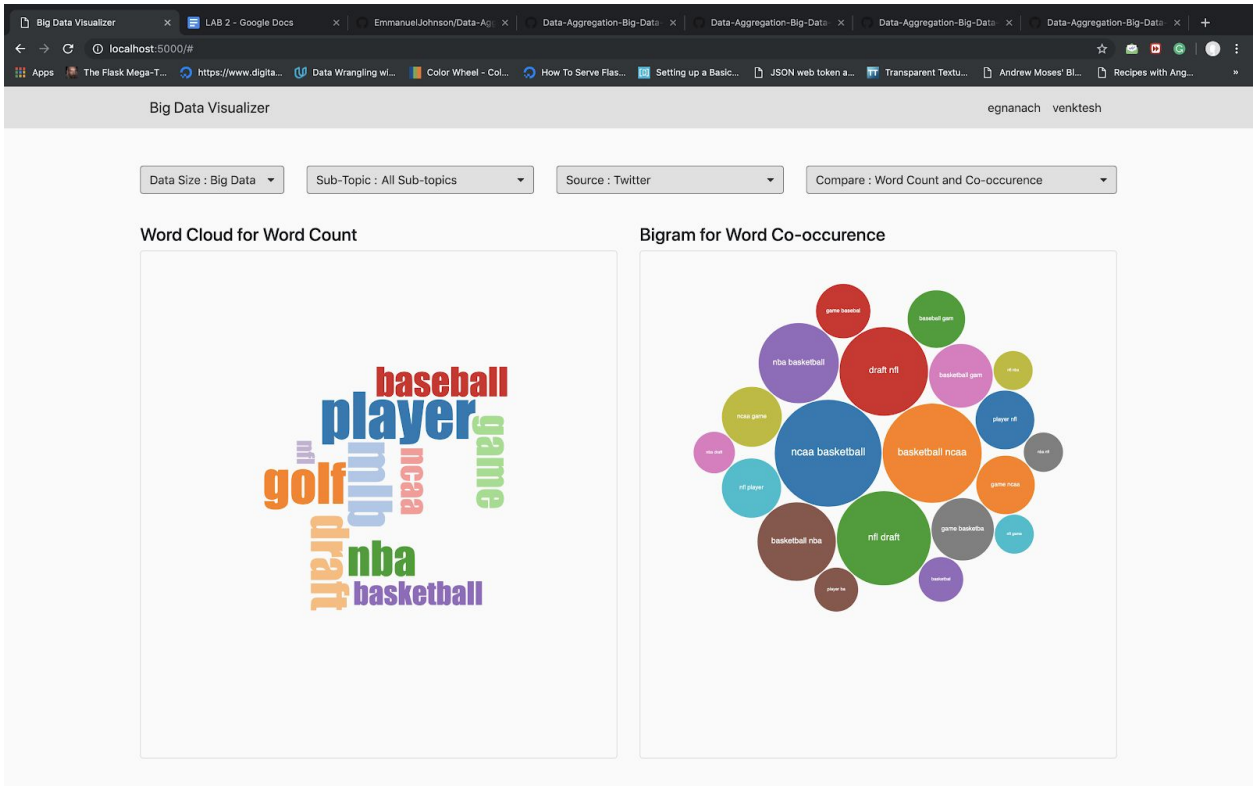
- This server.sh file is present inside the webpage folder as shown below,



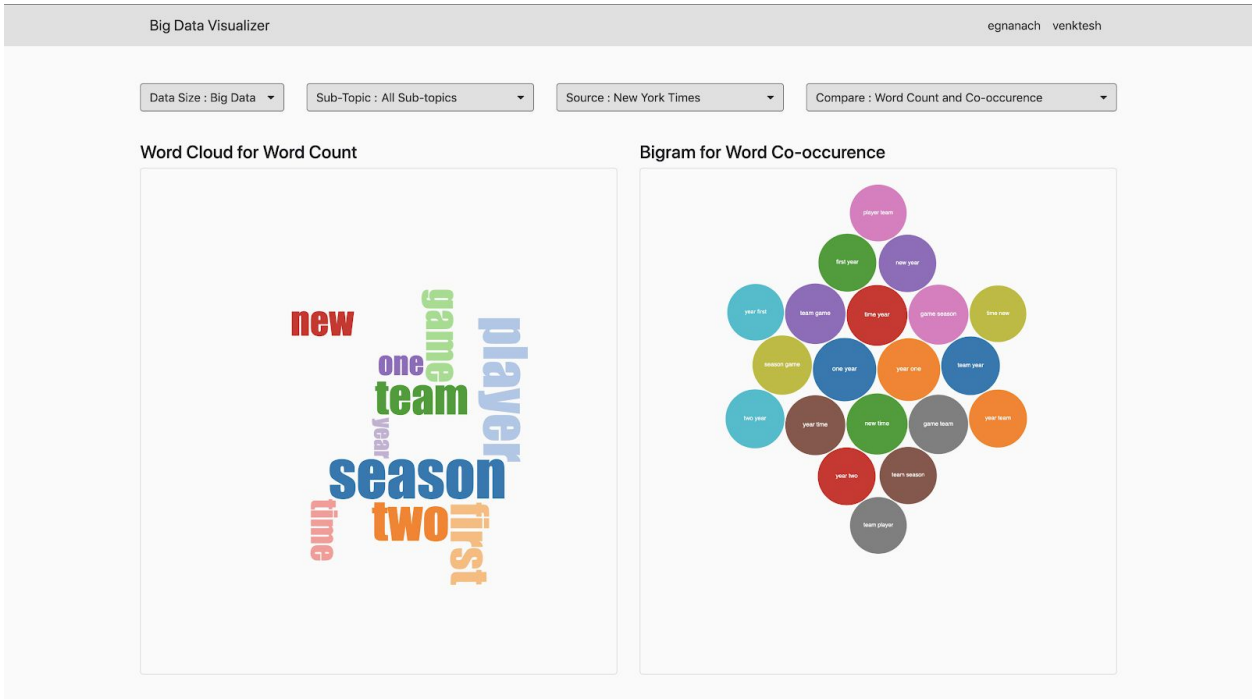


SCREENSHOTS :

Word Count and Word Co-occurrence for Twitter Big Data - All Subtopics

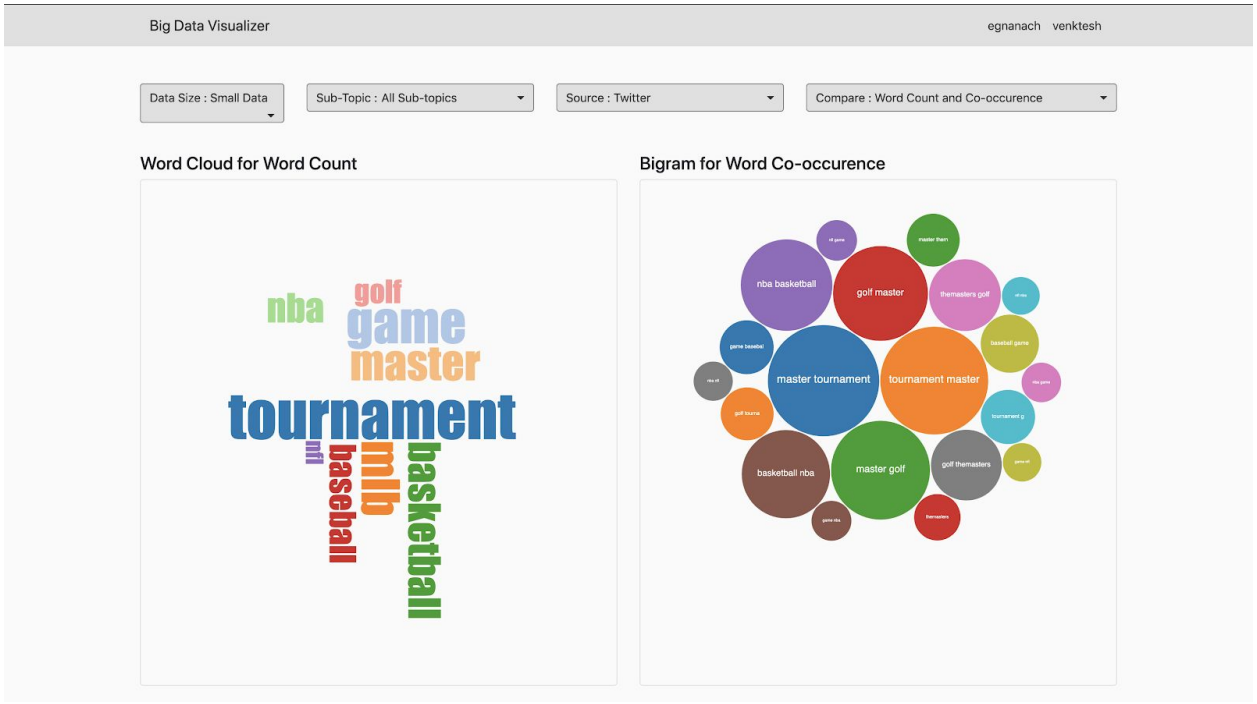


Word Count and Word Co-occurrence for New York Times Big Data - All Subtopics



Word Count and Word Co-occurrence for Common Crawl Big Data - All Subtopics

Word Count and Word Co-occurrence for Twitter Small Data - All Subtopics



Word Count and Word Co-occurrence for New York Times Small Data - All Subtopics

Word Count and Word Co-occurrence for Common Crawl Small Data - All Subtopics