LAB2 – DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

Team Members:
Pratik Sanghvi
Shivangi Tripathi

This project helps us explore the entire data pipeline. We look at aggregating data from multiple sources using APIs, applying big data analytical method of MapReduce, store the data in a Hadoop infrastructure and build a visualization product using Tableau to generate insights.

American elections are something that's of importance to people all around the world. We've taken elections as a topic and tried to generate some insights out of it. Within ELECTION, we use sub topics: TRUMP, CAMPAIGN, IMMIGRATION, ECONOMY, HEALTHCARE and REPUBLICAN.

PART-1

Part 1 involves data collection. We collect data from 3 sources: twitter, NYTimes and common crawl.

For twitter:

- We collect the tweets for each sub topic using Rtweet package and save it in csv files
- Next, we combine all the csv files by sub topics
- We then append all the sub topics and create a final text file
- Then, using the NLTK and Re package, we apply regex on data, tokenize all the words and then remove stop and stem words and store the output in a text file
- For, stemming, instead of using stemmer we use lemmatizer[4 from Reference]
- Total number of tweets: 29722

For NYTimes:

- We used the nytimes API and extracted the 100 urls for each sub topic. In total, we extracted 600 urls.[7 from Reference]
- We then used the Beautiful Soup package for web crawling and extracted the text from each url.
- Then, using the NLTK and Re package, we applied regex on data and tokenized all the words and then remove the stop and stem words and store the output for each subtopic in a text file.
- For, stemming, instead of using stemmer we use lemmatizer[4 from Reference].

For Common Crawl:

- We start by downloading wet.paths files, then we extract the urls for the data segments of S3 bucket
- Each url in wet.paths files, is a link for data segment in Amazon S3 bucket for common crawl.

- We append 1 url with https://commoncrawl.s3.amazonaws.com/ and download zip files containing data content in the format .gz
- Now, using the warcio package, we search our sub topics inside the .warc.wet.gz file and extract the data in a text file. [3 from Reference]
- Finally, we get 1 text file containing data for all sub topics

PART-2

Part-2 involves setting up the Hadoop infrastructure, testing the code and infrastructure on a small sample data.

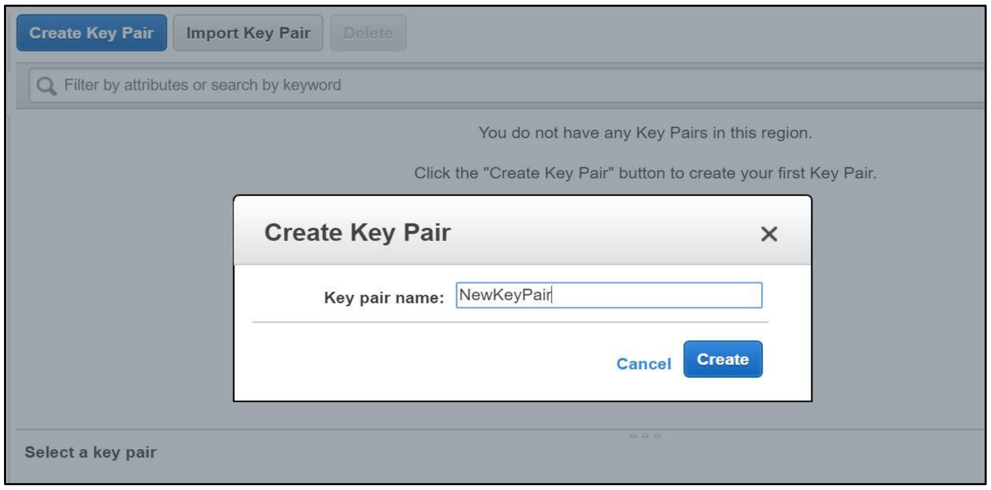
- We take pg345.txt as our sample data
- Clean this sample data by removing special characters, removing stop words, tokenizing and then lemmatizing the tokenized words. The output is then saved by the name filtered.txt
- We then set up our Hadoop cluster on using Elastic MapReduce
- Now, by uploading the filtered text file and the wordcount .jar file in S3 bucket, we run job and the output is stored in the output folder of S3 bucket [2,5 from Reference]

PART-3

Part 3 involves using the infrastructure set up in part 2 and running a MapReduce algorithm on data from part 1.

Part 3a:

- In this part, we execute the entire data pipeline for the entire dataset from all the 3 sources
- Again, we start by cleaning small data by removing special characters, removing stop words, tokenizing and then lemmatizing the tokenized words. All the cleaning is done in python
- Next, we then set up our Hadoop cluster on using Elastic MapReduce
- Following screenshot gives a visual representation of the entire MapReduce algorithm
- Start by creating a key pair



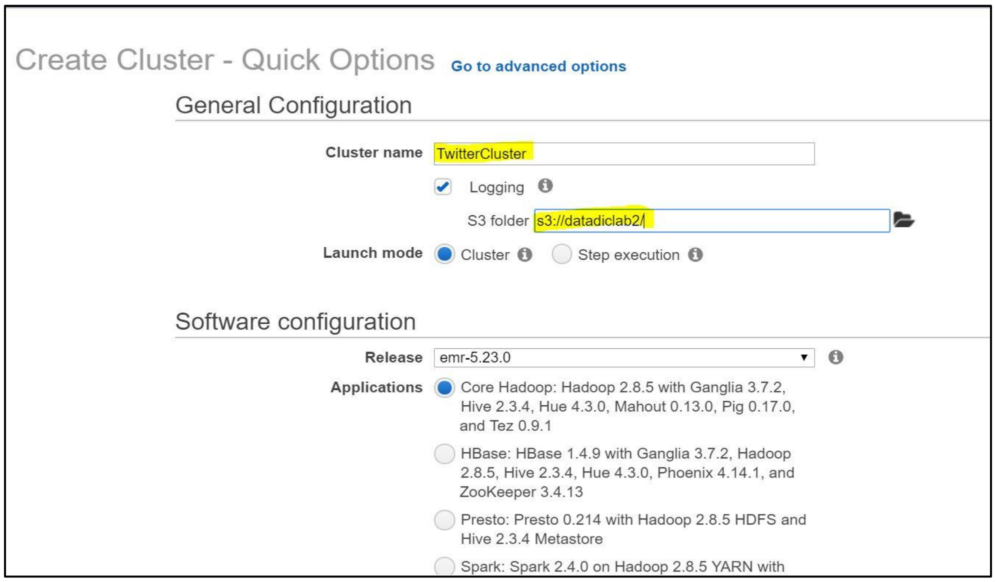
You do not have any Key Pairs in this region.

Click the "Create Key Pair" button to create your first Key Pair.

Create Key Pair

Key pair name: NewKeyPair

- Next, create a cluster in EC2



Create Cluster - Quick Options Go to advanced options

General Configuration

Cluster name: TwitterCluster

[✓] Logging

S3 folder: s3://datadiclab2/

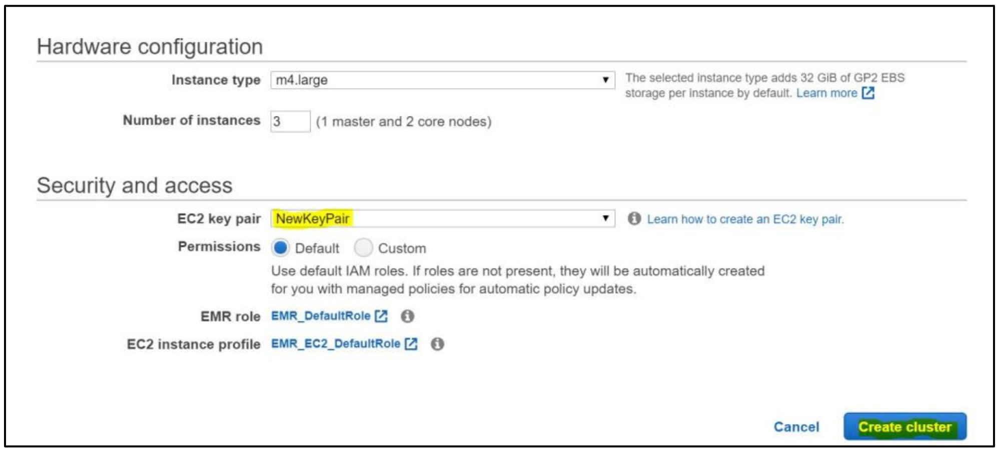
Launch mode: [✓ Cluster] Step execution

Software configuration

Release: emr-5.23.0

Applications:
- [✓ Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.4, Hue 4.3.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.1]
- HBase: HBase 1.4.9 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.4, Hue 4.3.0, Phoenix 4.14.1, and ZooKeeper 3.4.13
- Presto: Presto 0.214 with Hadoop 2.8.5 HDFS and Hive 2.3.4 Metastore
- Spark: Spark 2.4.0 on Hadoop 2.8.5 YARN with

- We then link the key pair with cluster and create a cluster



Hardware configuration

Instance type: m4.large — The selected instance type adds 32 GiB of GP2 EBS storage per instance by default. Learn more

Number of instances: 3 (1 master and 2 core nodes)

Security and access

EC2 key pair: NewKeyPair — Learn how to create an EC2 key pair.

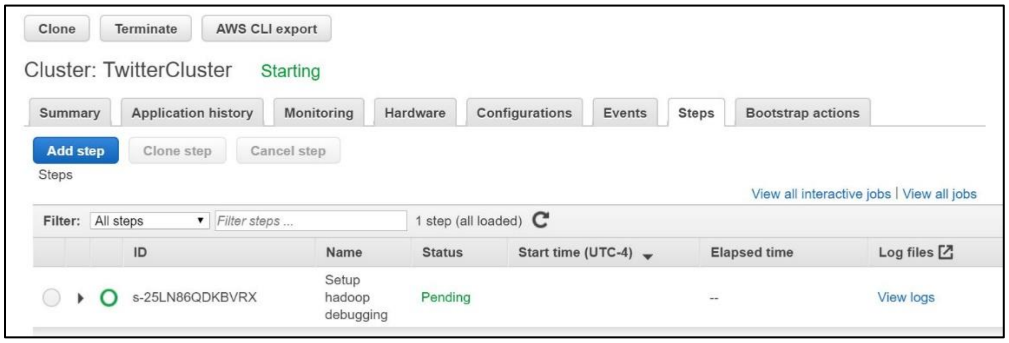
Permissions: [✓ Default] Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role: EMR_DefaultRole

EC2 instance profile: EMR_EC2_DefaultRole

- The cluster takes a few minutes to be created, once set up, its status changes from pending to running



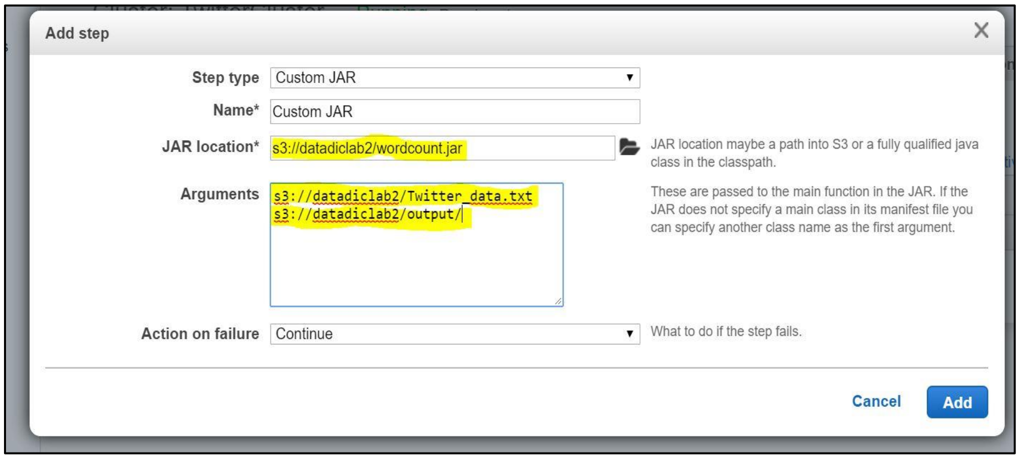
Cluster: TwitterCluster Starting

Steps

Filter: All steps | Filter steps ... | 1 step (all loaded)

ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files
s-25LN86QDKBVRX	Setup hadoop debugging	Pending		--	View logs

-
- Once the cluster has been created, we then add the input text file and MapReduce .jar file in S3 bucket and run a job



Add step

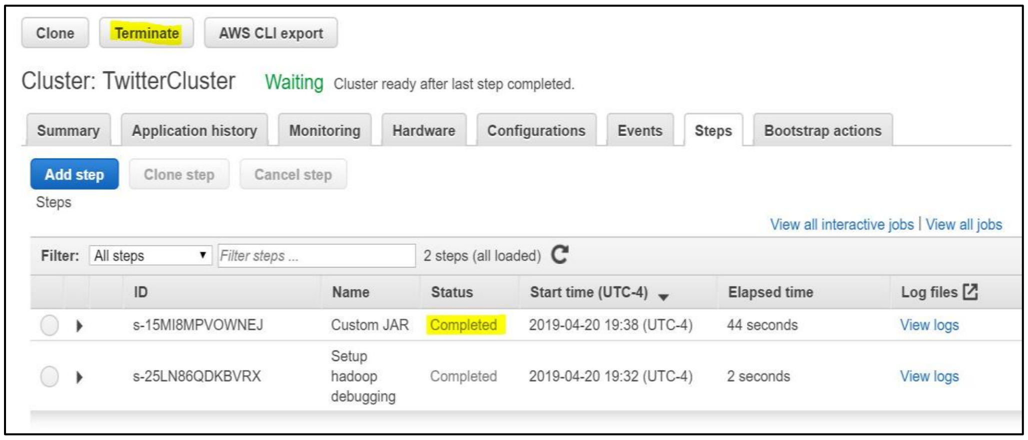
Step type	Custom JAR
Name*	Custom JAR
JAR location*	s3://datadiclab2/wordcount.jar
Arguments	s3://datadiclab2/Twitter_data.txt s3://datadiclab2/output/
Action on failure	Continue

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

What to do if the step fails.

-
- Once we click on add, the MapReduce algorithm runs and the output is stored in the output folder in S3 bucket. The status now changes to Completed



Cluster: TwitterCluster Waiting Cluster ready after last step completed.

Filter: All steps Filter steps ... 2 steps (all loaded)

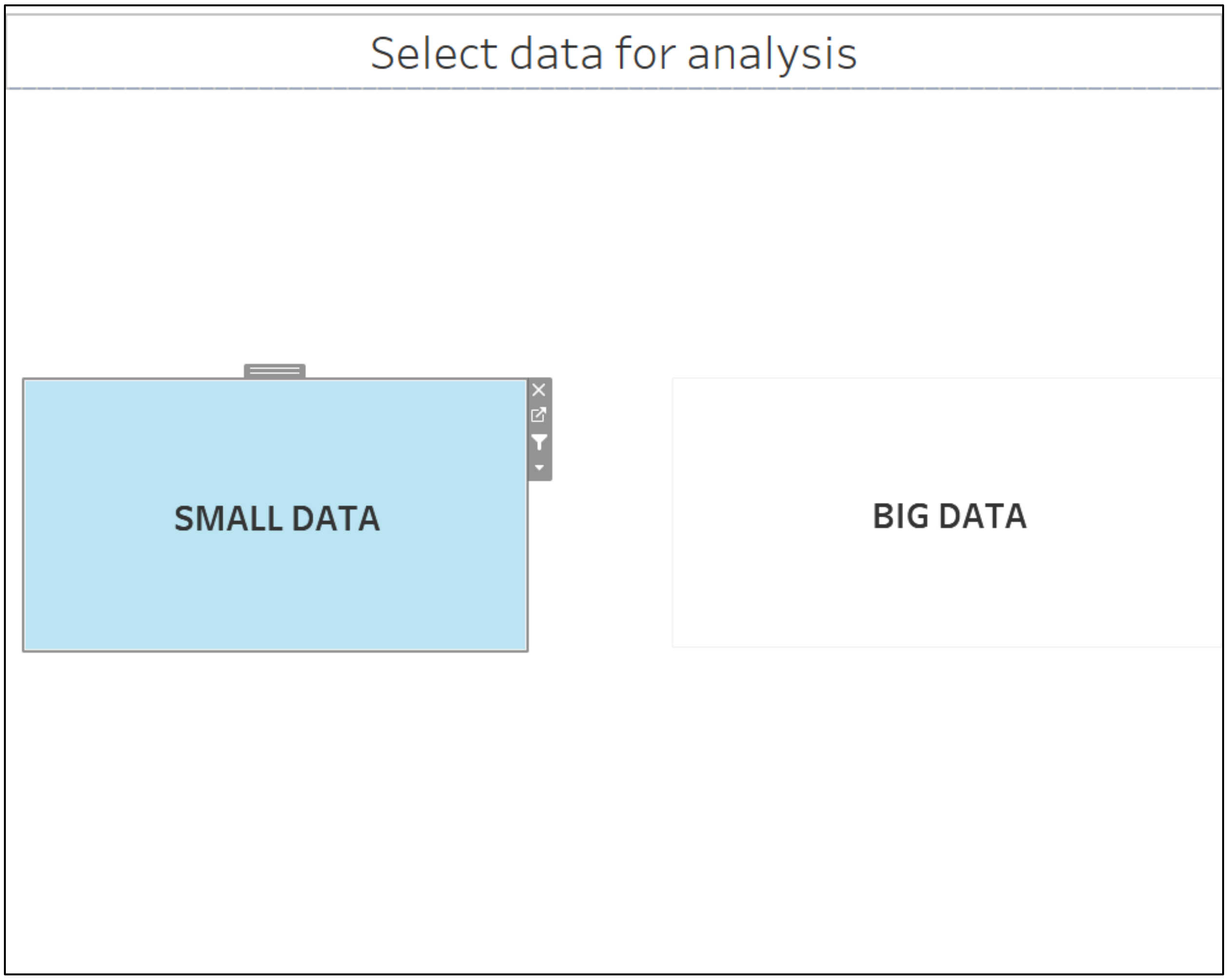
ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files
s-15MI8MPVOWNEJ	Custom JAR	Completed	2019-04-20 19:38 (UTC-4)	44 seconds	View logs
s-25LN86QDKBVRX	Setup hadoop debugging	Completed	2019-04-20 19:32 (UTC-4)	2 seconds	View logs

- The process for co-occurrence has been explained in part 3e.

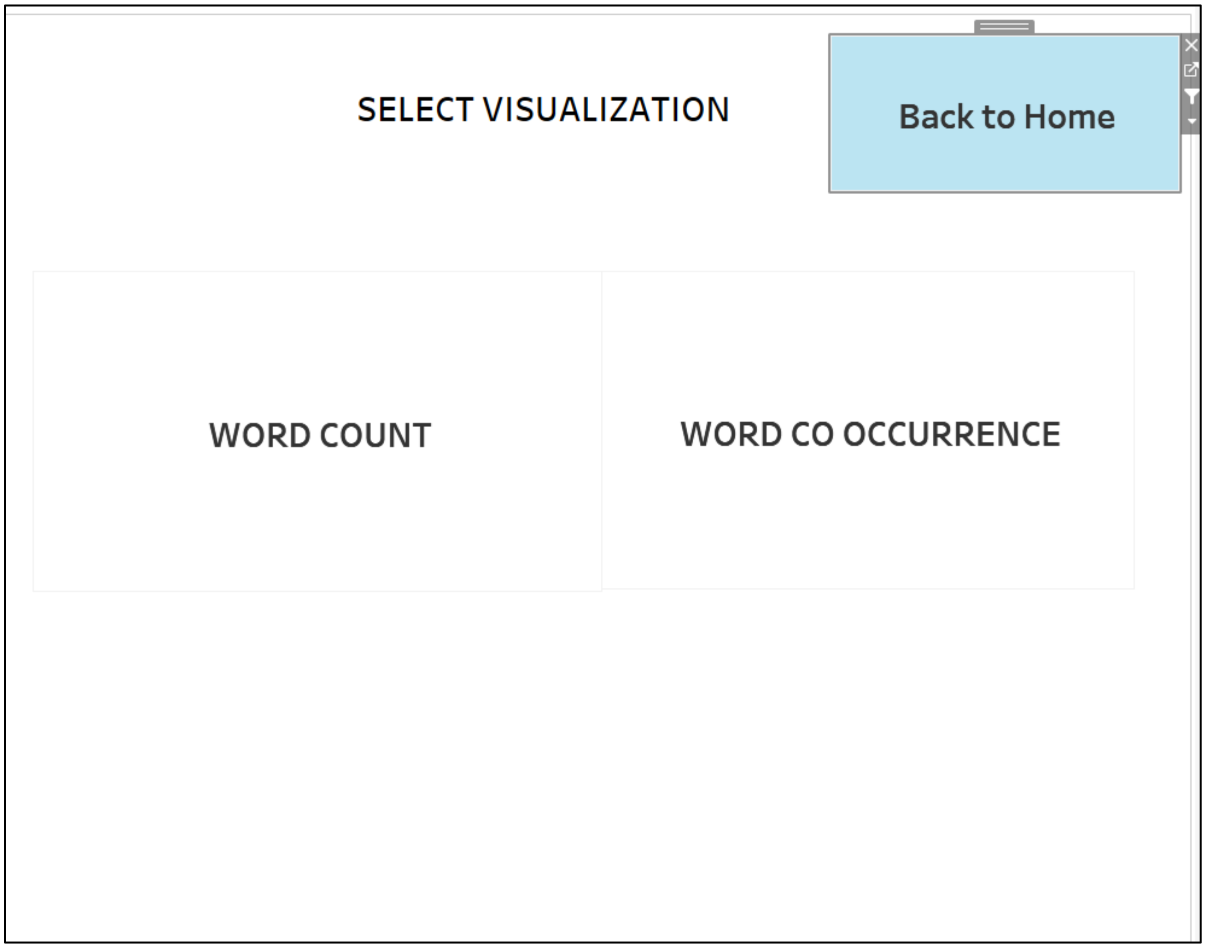
Part 3b:

In this part, we take the output from the MapReduce job and visualize it in tableau as a Word Cloud.

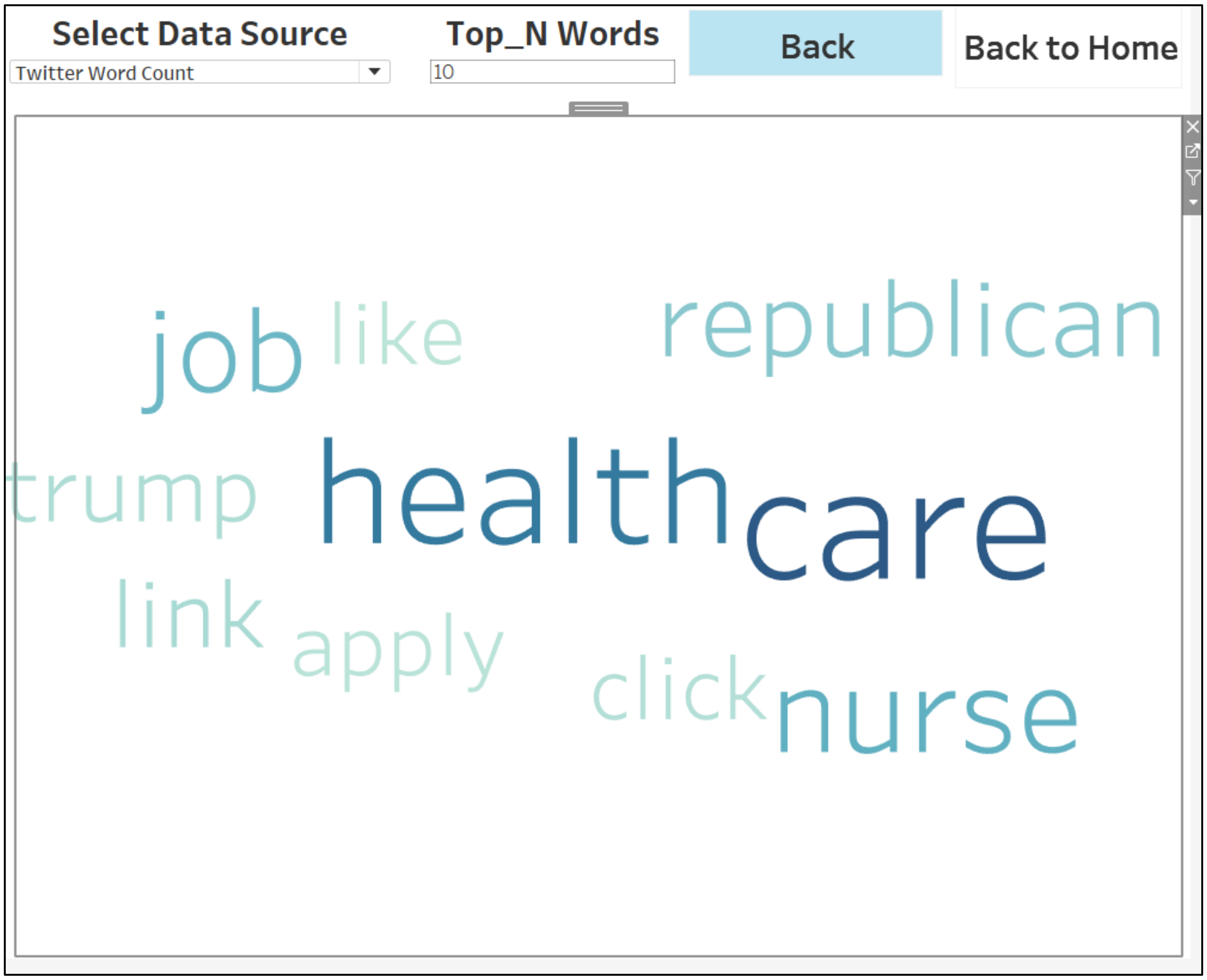
- We create a dashboard in tableau[6 from Reference]
- The home screen has 2 buttons for small and big data
- The small data has visualization for all 3 data sources for a small amount of data, while the big data tab has visualizations for all 3 data sources for the entire data:



- Selecting the small data would take us to the next screen:



- Now, based on our requirement, we click on either Word Count or Word Co-occurrence buttons. Clicking on Word count would then take us to the next screen:



- On the top left, we have the different data sources to select. Next we can select the top_n words that we want to visualize.
- We have a back button that can take us back to the previous screen and on the top right corner we have 'Back to Home' button that can take us back to the home screen

Part 3c:

In this part, we execute the entire data pipeline i.e. steps a to c, for a small dataset for 1 week, for all the 3 sources. The outputs for big data and small data are stored in part3 folder.

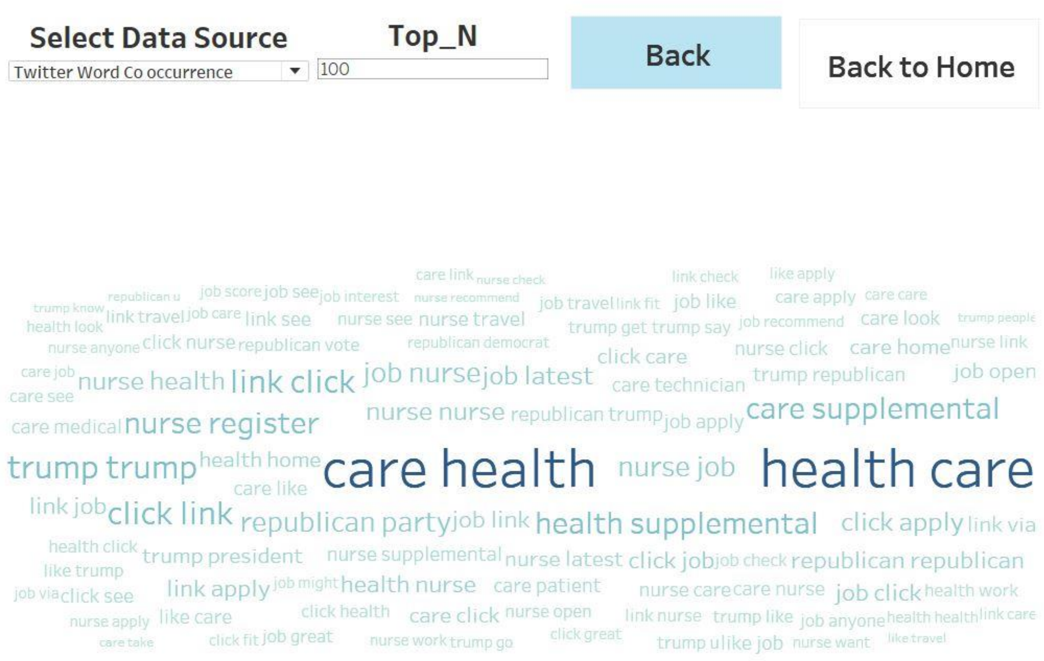
Part 3d:

For this part, we create a dashboard in Tableau, where we can see the word cloud for word count or word co-occurrence, for any of the data sources. We create different screens for small and big data and drop downs for different data sources. This is explained in detail using screenshots in part 3b.

Part 3e:

In this part, we use the output for word count, we take the top 10 words and use them in the word co-occurrence MapReduce .jar file. The entire process of creating a cluster and running the job is repeated for word co-occurrence. We then process the output to get pair for visualization in word co-occurrence using WordCoOccurrencePostProcessing.py and save this processed output in a csv. [1,5 from Reference]

The output of Word Co Occurrence comes like this:-



References:-

1. Co Occurrence Java Program:-
    a. https://dzone.com/articles/calculating-co-occurrence
    b. https://www.javacodegeeks.com/2012/11/calculating-a-co-occurrence-matrix-with-hadoop.html

2. Word Count Java Program:-
    a. https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

3. WARC File Reader Library For Common Crawl:- https://github.com/webrecorder/warcio
4. Lemmatization:- https://www.machinelearningplus.com/nlp/lemmatization-examples-python/
5. Jar Reference:- http://tuttlem.github.io/2014/01/30/create-a-mapreduce-job-using-java-and-maven.html
6. Tableau:- https://towardsdatascience.com/word-clouds-in-tableau-quick-easy-e71519cf507a
7. NYTimes Data Collection:- https://dlab.berkeley.edu/blog/scraping-new-york-times-articles-python-tutorial