DIC LAB – 2 REPORT : DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

Muthuvel Palanisamy – 50246815

Akshay Chopra – 50248989

TOPIC :

“Facebook and Cambridge Analytics Data Breach”

**Part – 1: Data Collection**

* R is used as the language for data collection and cleaning
* A simple block diagram is shown in figure 1

**NY TIMES articles:**

* **Rtimes** packages is used to extract the url of articles using keyword like “*cambridge annalytica”, “ facebook scandal”, “facebook data leak”…*
* **Contentscrapter in Rcrawler** package is used to crawl the webpages containing the url and extract the article content
* A total of around 250 articles is present in the data collected after removing duplicated articles and articles that does not belong to the topic
* The Articles collected is present in directory Part2/Data/articlesTotal.txt

**TWITTER tweets:**

* A total of 10,500 tweets is collected on the topic using the keyword described above and hashtags like *“#deletefacebook”*
* **TwitteR** package is used to extract tweets
* Tweets are filtered using the text, tweet ids to remove duplicates
* Data is cleaned by remove non-ASCII characters, symbols as preprocessing step
* The tweets collected in present in Part2/Data/tweetsTotal.txt

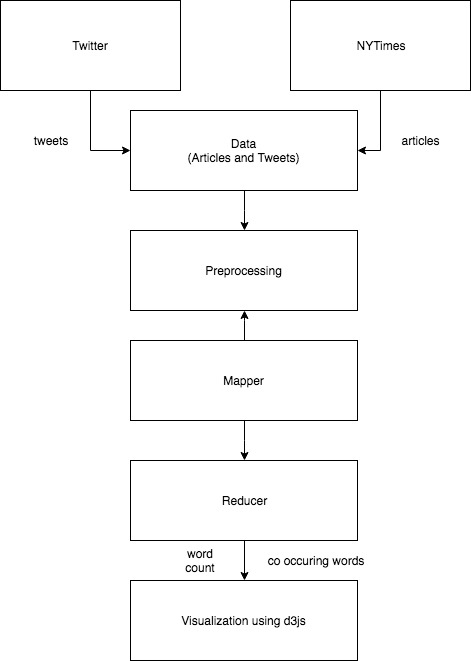


Figure 1: Block diagram of our implementation

**Part – 2: Word Count using map reduce in Hadoop**

* Python is used for creating the mapper and the reducer scripts
* Seprate mapper and reducer scripts are written for both single word count and co-occurring word count. So, for single word count there is a mapper and reducer and for co-occurring words also there is a mapper and reducer.

**mapper.py**

* The mapper.py file outputs the words with count as 1 as <key, value> pair for single word count part and co-occurring words with count as 1 as <key, value> pair for co-occurring word count part.
* NLTK library of python is used for creating list of stop words and for word tokenizing. Additionally, we have also included a list of our own stop words (which we couldn’t find in the NLTK package).
* So, stop words are stopwords of NLTK for “English” language and list of our own stop words.
* Additional list of symbols is also created so that we can ignore the word if it’s a symbol.
* Ultimately, if a word is a symbol or a stop word or of length 1, it is ignored and not outputted from the mapper (Word of length 1 is also ignored since it won’t make any sense to output word of unit length).
* Output of the mapper is sorted according to the key. The key and value (single word/ co-occurring words and ‘1’) are tab separated.

**reducer.py**

* The reducer splits the output of the mapper based on tab ‘\t’.
* It then checks that if the keys are same (single word/co-occurring words), it adds their corresponding values.
* The reducer outputs the word/co-occurring words with its actual count.

**extract\_top50\_singlewords.py**

* This file takes output of the reducer (single words with their count), sorts it according to the count and outputs top 50 words based on the count to a csv file.
* The csv file is ultimately used as data for visualization of the word cloud for single words.
* We used top 50 words and not top 10 since if we used only top 10, the word cloud would look sparse.

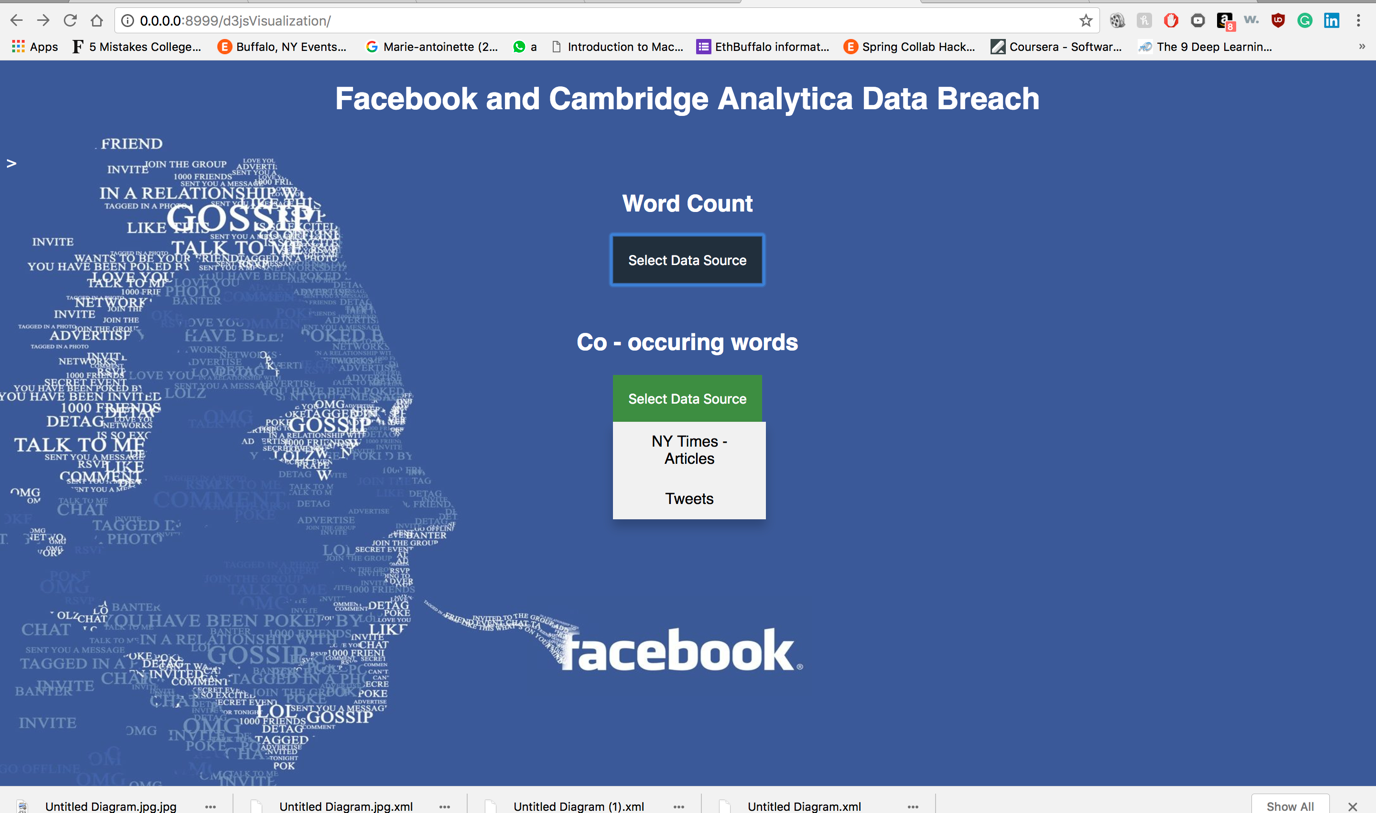
**extract\_top10\_cooccurence.py**

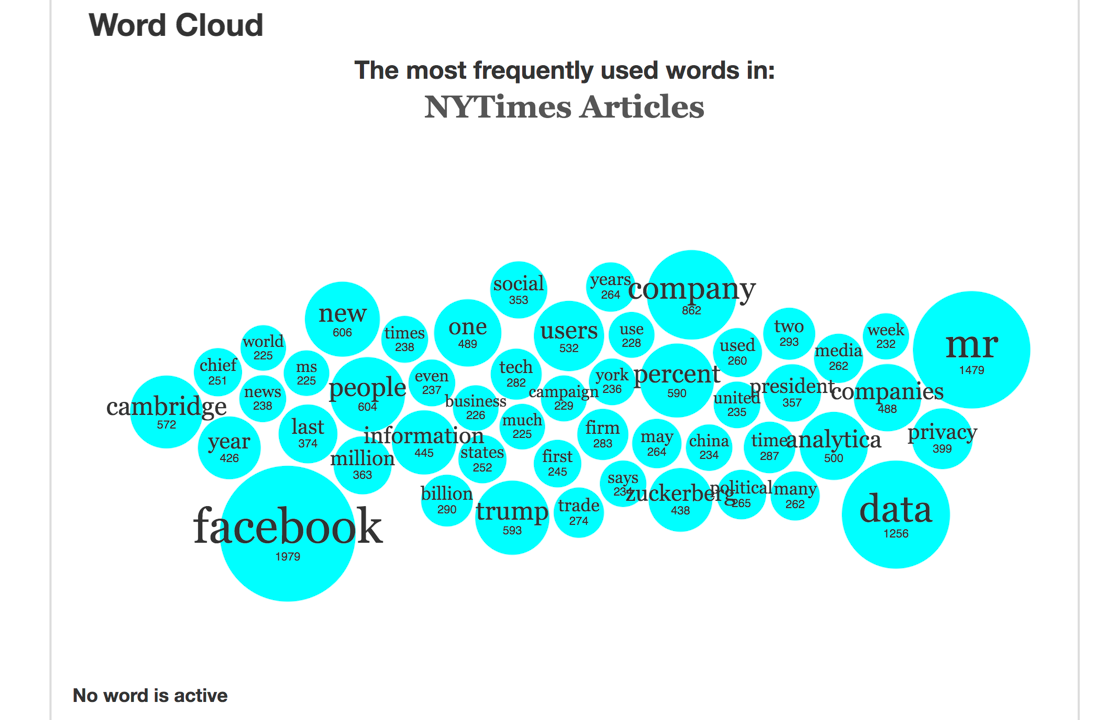
* The file takes output of the reducer (co-occurring words with their count), sorts it according to the count and outputs top 10 words based on the count to a csv file.
* The csv file is ultimately used as data for visualization of word cloud for co-occurrence.
* We took top 10 since it was mentioned in the lab pdf.

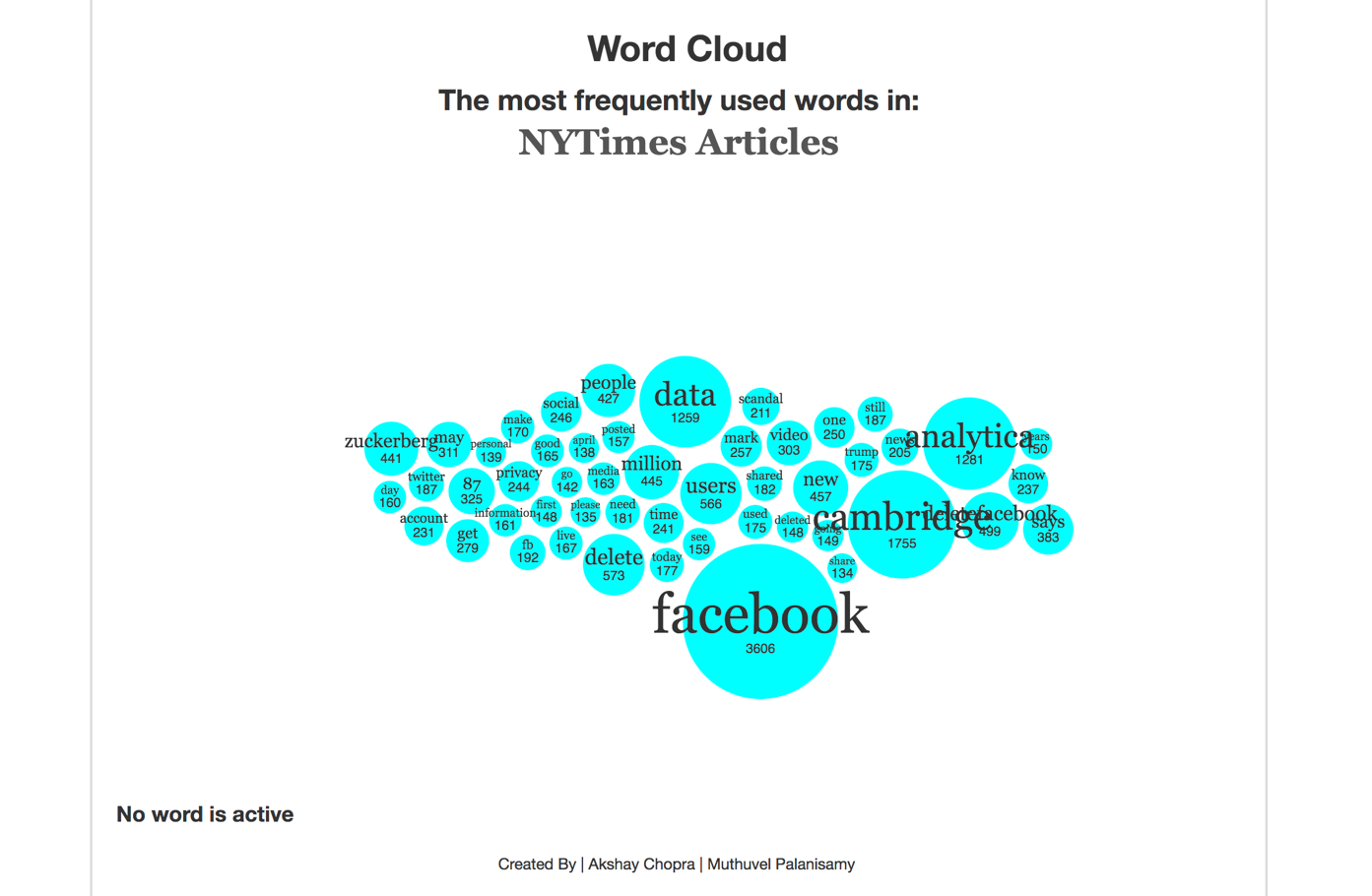
**Part – 3: Visualization of word count**

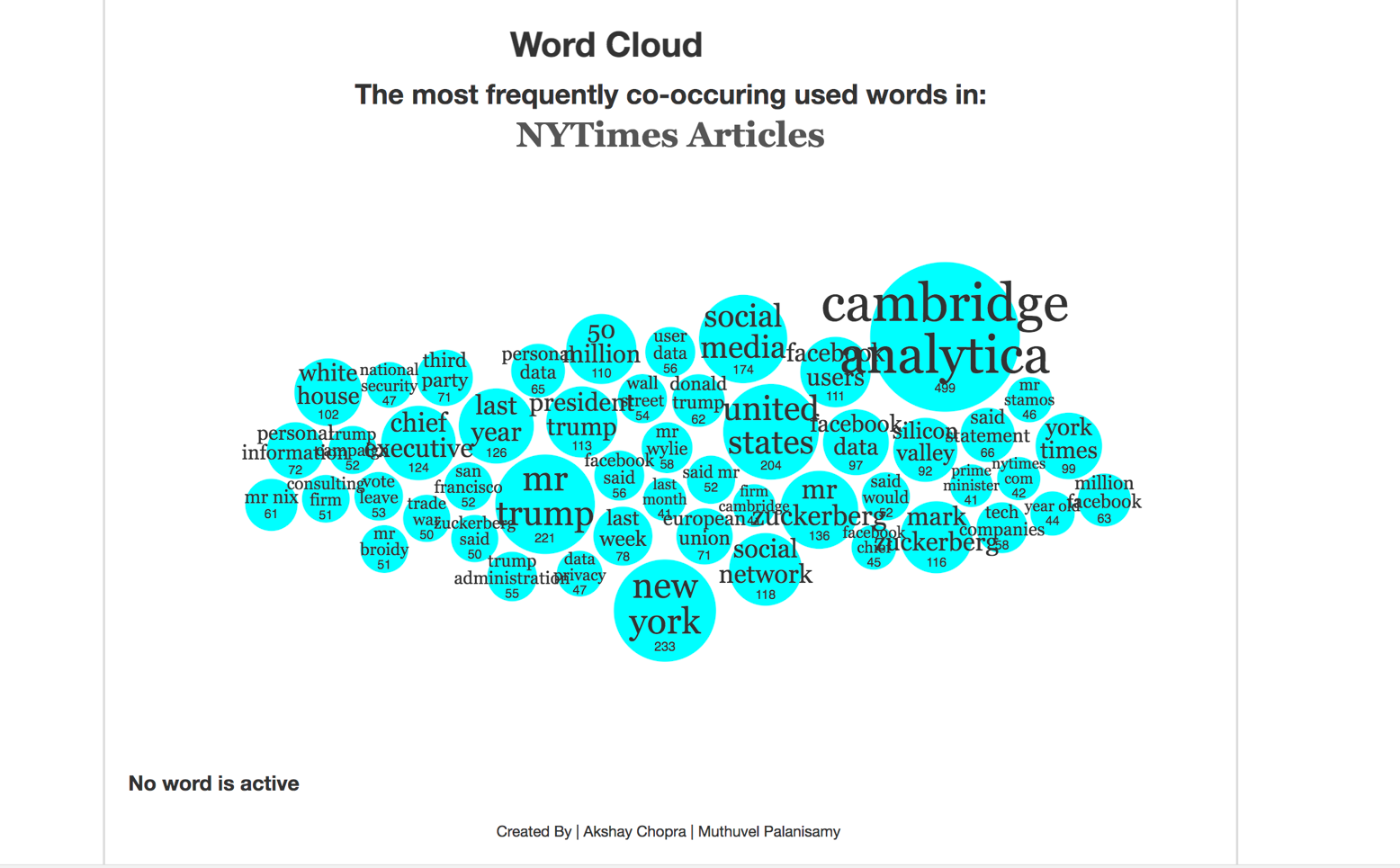
* Used **D3.JS** to visualizing the word cloud for the **top 50** word count and **top 10**  co- occurring words, adapted from the source code provided in [1].
* The output is visualized in a webpage as follows
* Each blob visualized displays the count and the word/co-occurring word, with the size of the blob depending on the count

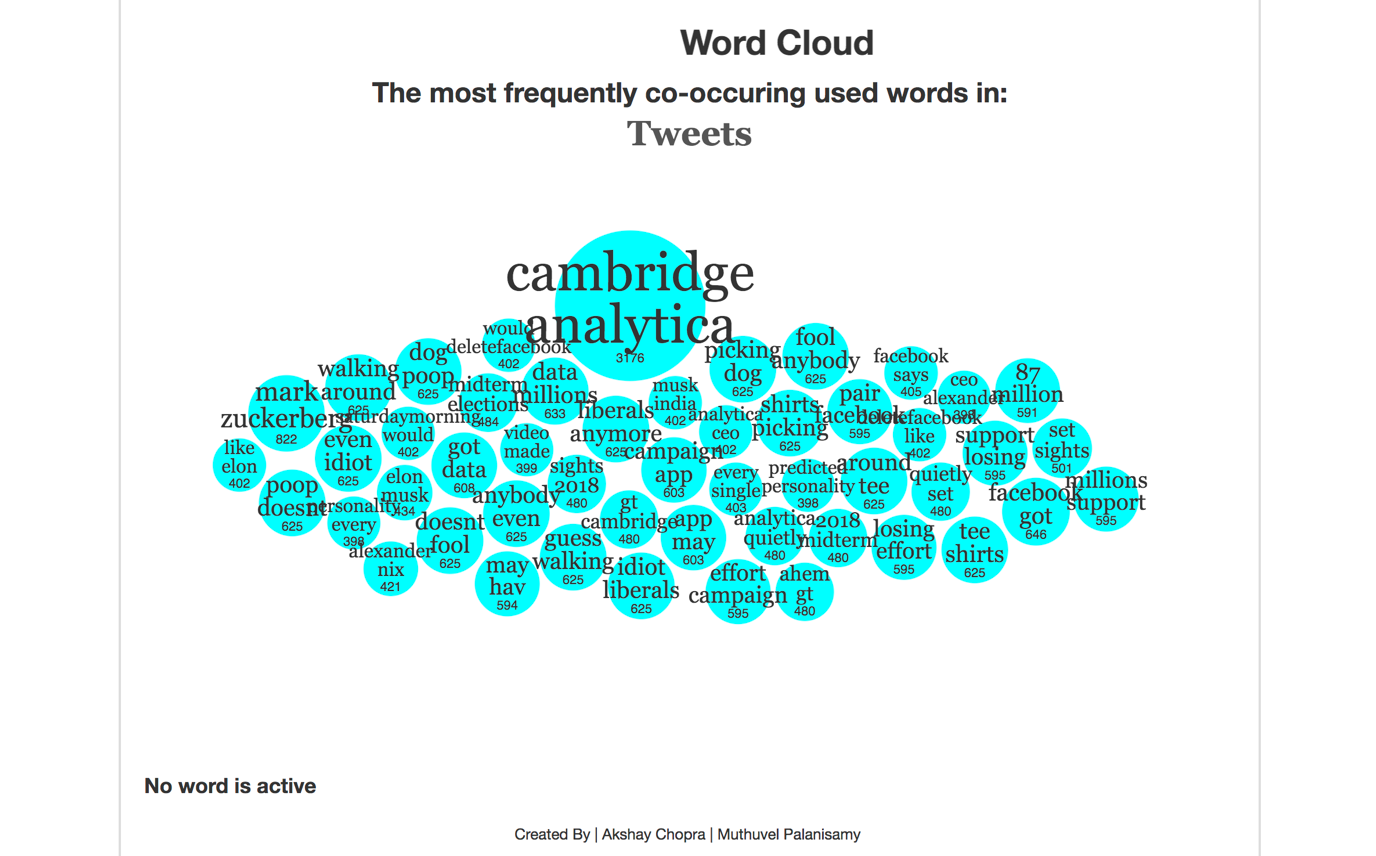
Homepage:

****









**Analysis**

* Words like “cambridge”, “analytica”, “facebook”, “Zuckerberg”, “data” were having very high frequency and were found to be common in both tweets and articles data

**References:**

[1] [https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html - Reducer](https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Reducer)

[2] <http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>

[3] <https://github.com/vlandham/bubble_cloud>