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

Muthuvel Palanisamy – 50246815
Akshay Chopra – 50248989

TOPIC :
“Facebook and Cambridge Analytics Data Breach”

INSTRUCTION FOR TESTING THE IMPLEMENTATION:

- Make the working directory to source file location before testing any code or you will get error because relative paths
- Create a Python server using “python3 –m http.server” to create a local server and run the visualization from there for d3js

Directory paths:

For testing data collection:

- */Part2/code/dataCollection/tweetCollectorScript.**R*** -> to collect tweets and store them
- */Part2/code/dataCollection/nyTimesArticleCollectorScript.R* -> to collect article urls on the topic and store them
- */Part2/code/dataCollection/articleExtraction.R* -> to extract the content of the article from urls collected
- */Part2/data/tweetsTotal.txt, /Part2/data/tweetsOneDay.txt* -> tweets collected and stored for Map Reduce
- */Part2/data/articlestotal.txt, /Part2/data/articlesOneDay.txt* -> articles collected and stored for Map Reduce

For Map Reduce:

- */Part2/code/Hadoopcode* -> contain mapper and reducer files for Single Word Count
- */Part2/code/Hadoopcode* -> contain mapper and reducer files for Co-occurring Words Count

For d3js Visualization of word cloud:

- */Part2/code/d3jsvisualization/index.html* -> Run this file and choose the required input to check the word could for the specified input source

Video Demonstration link:

https://youtu.be/ZWIRwhqS8Y8

IMPLEMENTATION

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 dataleak"…*
- **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

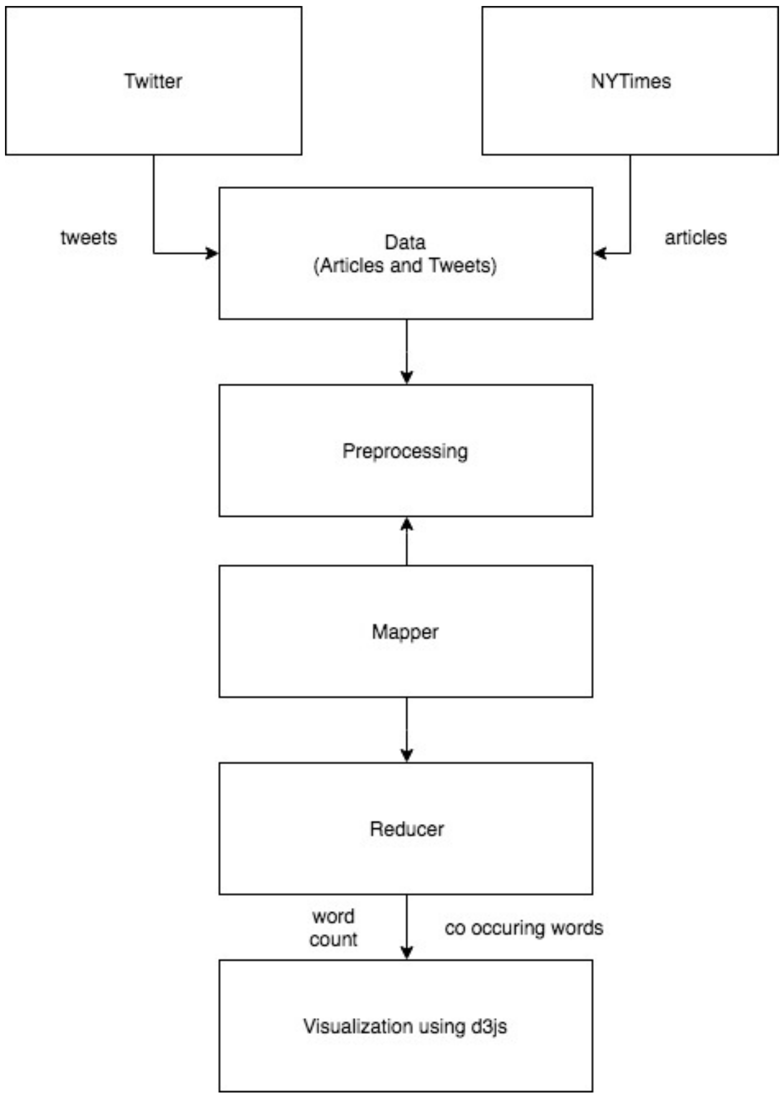


Figure 1: Block diagram of our implementation

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.

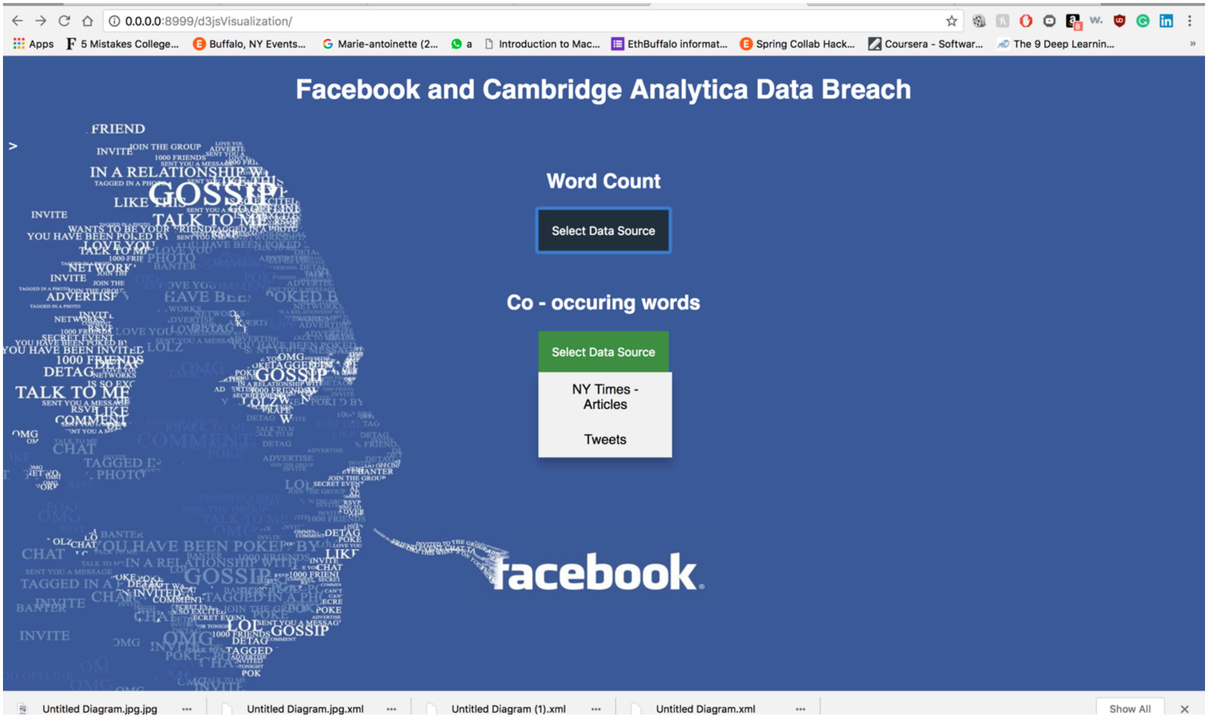
extract_top10_cooccurence.py

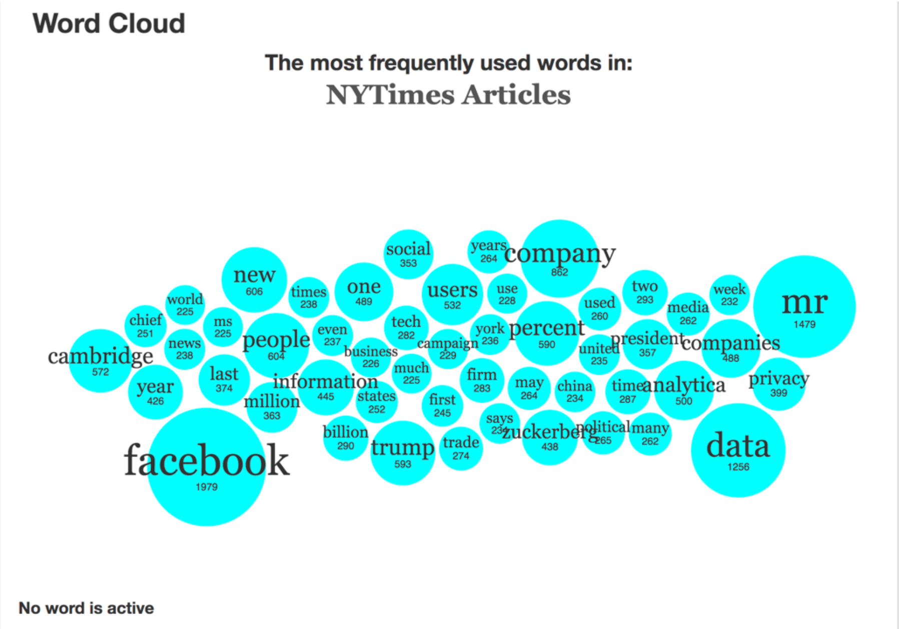
- The file takes output of the reducer (co-occurring words with their count), sorts it according to the count and outputs top 10 words based on the count to a csv file.
- The csv file is ultimately used as data for visualization of word cloud for co-occurrence.
- We took top 10 since it was mentioned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



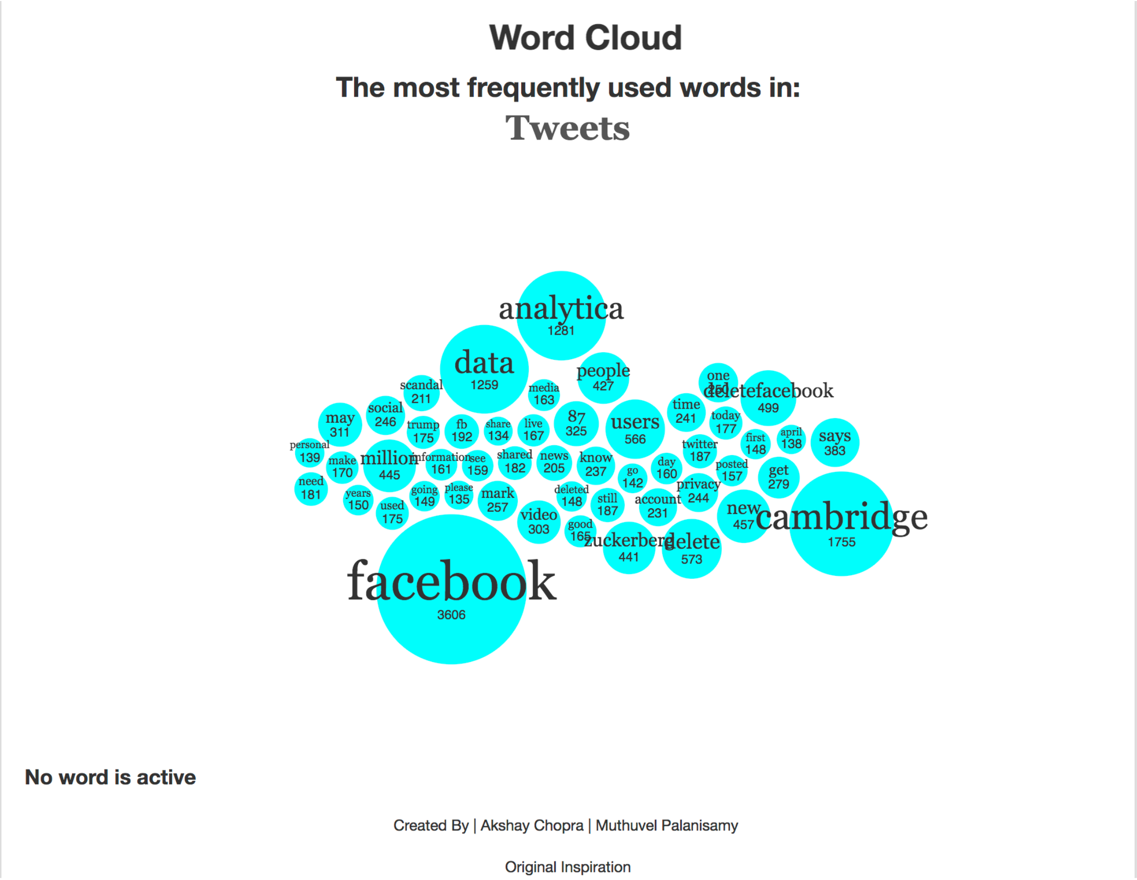


Word Cloud

The most frequently used words in:

NYTimes Articles

No word is active



Word Cloud

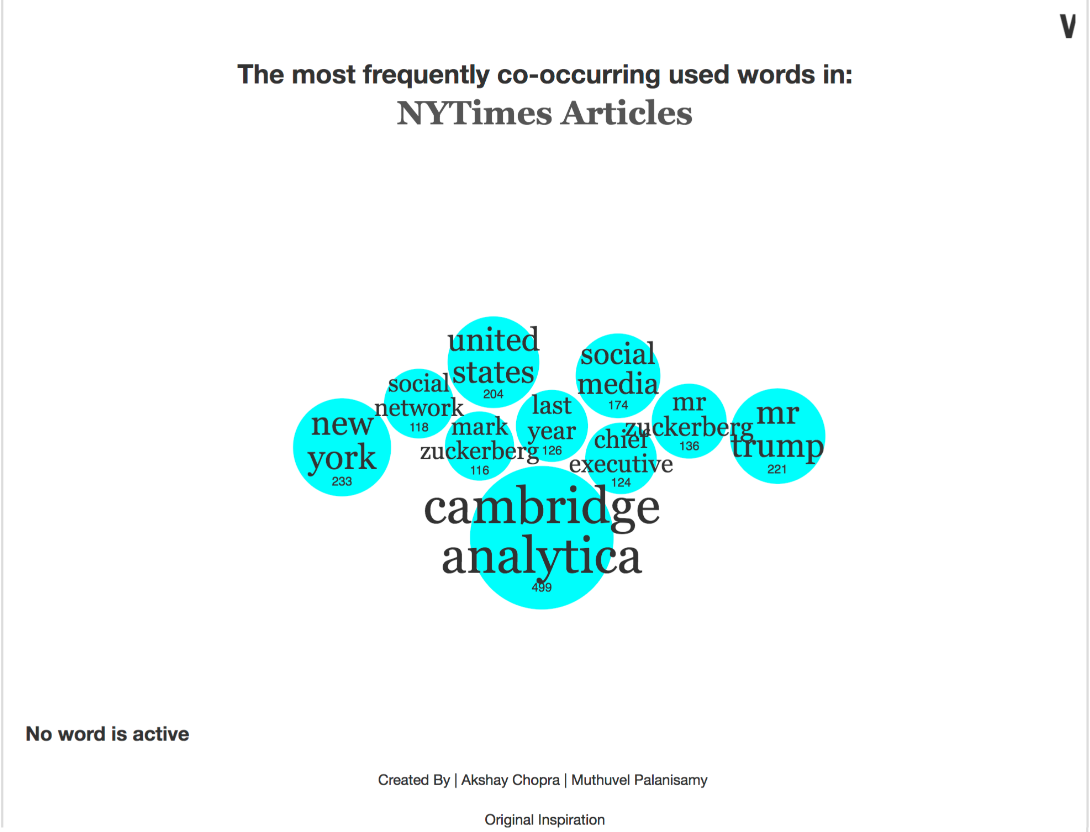
The most frequently used words in:

Tweets

No word is active

Created By | Akshay Chopra | Muthuvel Palanisamy

Original Inspiration



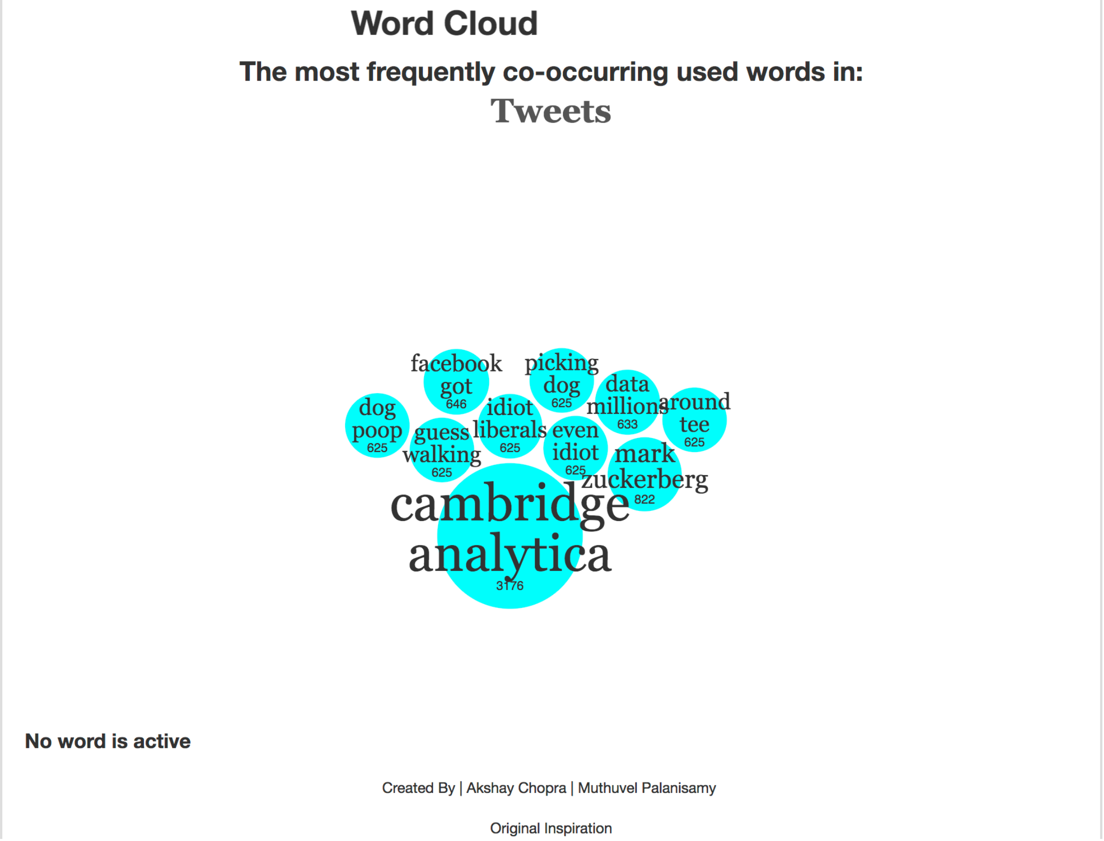
The most frequently co-occurring used words in:

NYTimes Articles

No word is active

Created By | Akshay Chopra | Muthuvel Palanisamy

Original Inspiration



Word Cloud

The most frequently co-occurring used words in:

Tweets

No word is active

Created By | Akshay Chopra | Muthuvel Palanisamy

Original Inspiration

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
- The word count on tweets show that “Facebook”, “Cambridge”, “analytica”,”data” – we presume the reason for this to be the common usage of these words in many tweets and they represent the idea on topic mostly
- Tweets also contain a lot of commonly used/ colloquial words that people use in their day to day life while the words in NYTimes articles are more refined
- Filtering and preprocessing tweets were quite harder because unlike NYTimes articles, they contained more non – ASCII characters, symbols and garbage words and removing them appeared to be tricky
- “Cambridge Analytica” is the most co – occurring word. This is because the company name is used almost many tweets or an articles in NYTimes.
- Some tweets like “dogs” and “idiot” to express their having high frequence denoting people using it to express their anger.

References:

[1] 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