TEAM MEMBERS:

Piyush Surendra Saravagi 50246596
Deep Narayan Mishra 50245878

Aggregation, Analysis and Visualization of Tweets and NY Times Articles for Gun Violence in the United States

Introduction/Purpose

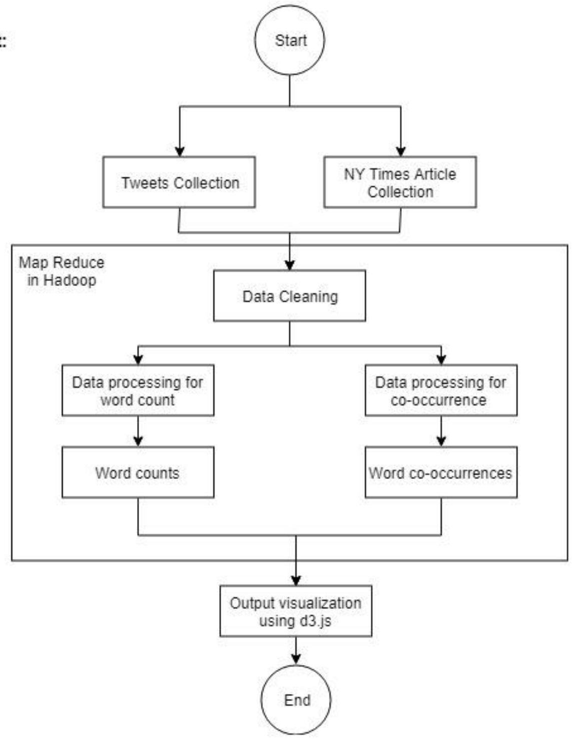
Mass shootings and school shootings have been in the spotlight of 2018 in United States. The most recent Parkland shooting student shook the entire United States and led to demonstrations in support of tighter gun control (March for Our Lives). Prominent figures took Twitter to express their opinions freely on the matter. A lot of criticism and reactions are seen from the masses. This makes gun violence an interesting topic for research to find statistics about what people are concerned the most.

Implementation

We start by gathering the tweet from Twitter and articles from NY Times. We then clean the data to separate the meaningful data from the noise. We process the data to find the most used words and their co-occurrences with each other. We then visualize the results to bring out the crux of the data from the articles and the tweets. We collect the data in two different sets:

- Data for single day (March 20, 2018)
- Data for the week (March 21, 2018 – March 28, 2018)

Process flow for the project:



Twitter Data Collection:

To gather the tweets on related topic, we used twitteR API of Twitter. We used R Script to achieve this. The Script makes use of API Key and return tweets for given 'search query' and 'date range'. We have used following hashtags/search strings to collected tweets from united states.

#gunviolence
#gunsense
#guncontrol
#stopgunviolence
parklandshooting
#ParklandSchoolShooting
#FloridaSchoolShooting
floridashooting
#gunlaws
#noguns
#gunreform

The script uses the tweets' screenName to retrieve the user location and stores all the information about tweets into a csv file (for example, twitter_data_24.csv). It further extracts the tweets' text and store them into the text file (for example, twitter_data_24.txt).

NY Times articles data collection:

To gather the articles, we used the API provided by NY Times. We used Python for achieving this. Using the API Key, search keywords, and date, we gather the URL of the articles and store it. The response of the API call is in JSON format. We parsed the response to extract only the useful information (URL of the articles) while discarding the rest of the information. We store the URLs of the articles in two different files. One file contains the URLs of articles for single day while other file stores the URLs for the entire week.

After storing the URLs in a file. We scrape the articles using the links. Again, we use Python for achieving this. We used the library BeautifulSoup to get the HTML Page of the articles. We studied the structure of the HTML page returned to identify the body of the article. We then parsed the HTML page to get the body of the article. We store this information in a file. Each article is stored in a different file so that we can efficiently use MapReduce framework to process these files. Similar to the URLs, we store the articles in two sets, one for single day and another for the entire week.

Word Count using the MapReduce Framework

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. We use the MapReduce framework to count the number of times a word occurred in the tweets or articles. We do this to find the most frequent words which capture the essence of the topic. Mapper and Reducer are implemented as follows:

- Mapper: In the Mapper, we clean the data to keep only the words essential for our analysis. We remove stop words to make the data meaningful. Stop words are natural language words which have very little meaning, such as "and", "the", "a", "an" etc. These are commonly occurring words and will distort our analysis.

  The Mapper them emits (outputs) a key value pair for each word in the article or tweet. The key in this case is the word and value is 1. We will later, in Reducer, aggregate these 1 for every key (unique word) to find the count of that word.

- Reducer: Here we find the actual counts. Output of the mapper is fed as an input to the Reducer. The reducer collects <key,value> pairs which have the same key. The <key, value> pair are of the type <word, 1>. It then sums over all the values received for this key. This generates the count for that key. The reducer then emit (output) this value.

Co-occurrence using the Map Reduce Framework

We find out the co-occurring words in each article/ tweet. The aim is to find the pair of words which occur together most frequently. For this we use the top ten most frequently occurring words captured after running the Word Count on tweets and articles. The context for co-occurrence is chosen as a tweet or an article. The approach followed is similar to the one used in word count using Map Reduce. The Mapper and Reducer are implemented as follows.

- Mapper: The first step of the mapper is to clean the data (just like in Word Count). We do this by removing the stop words and other common words which do not reflect the data. We then select pairs of words from the tweet/article, such that at least one of those words lies in the top ten words which we have identified. This word-pair, consisting of two words, is emitted by the Mapper in <key, value> format as <word-pair, 1>

- Reducer: In the Reducer, we collect the all the <key, value> pairs that are emitted by the Mapper. All the values (1) associated with the same key (word-pair) are aggregated to get the count of the word-pair. This is emitted by the Reducer as the output. This procedure is applied to every unique word-pair sent as the key. The resulting output gives us the frequency of co-occurrence of the word-pairs so that we can identify the co-occurring words with the highest frequencies.

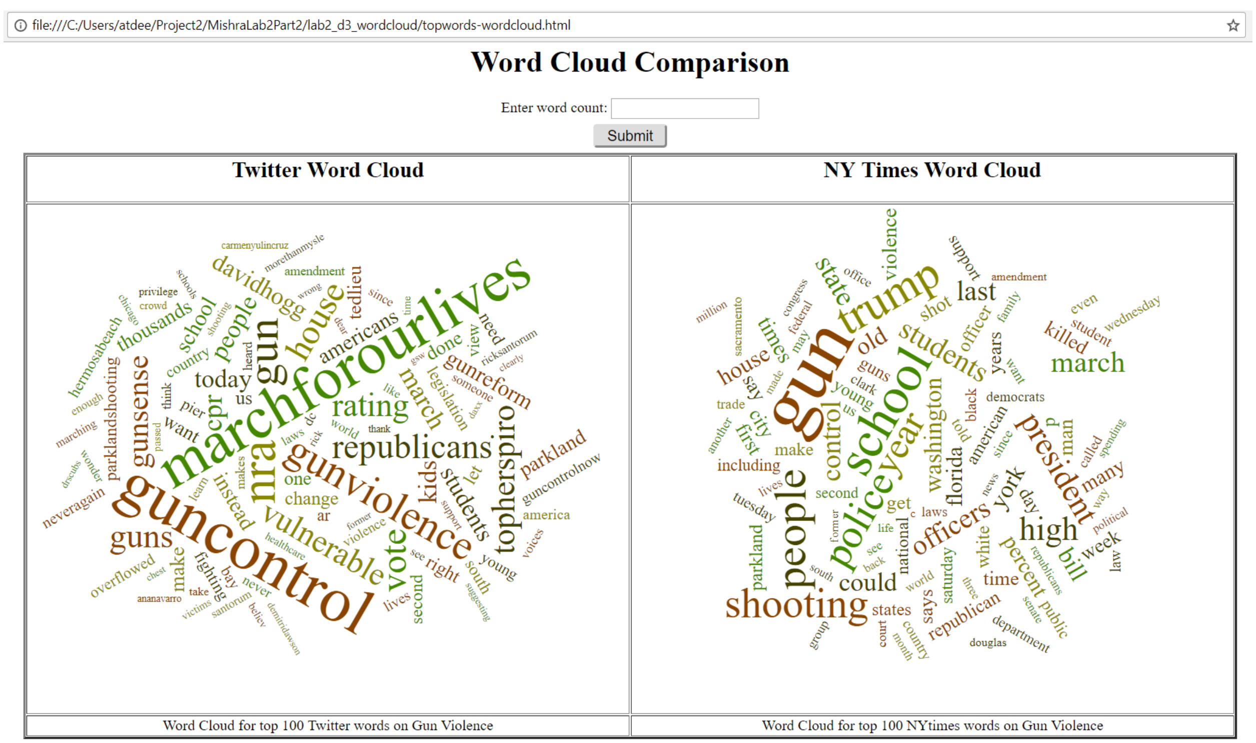
Visualizing the output using D3.js

To show the comparison of top occurring words in Twitter data vs NYTimes articles, we have used d3 word cloud visualization.

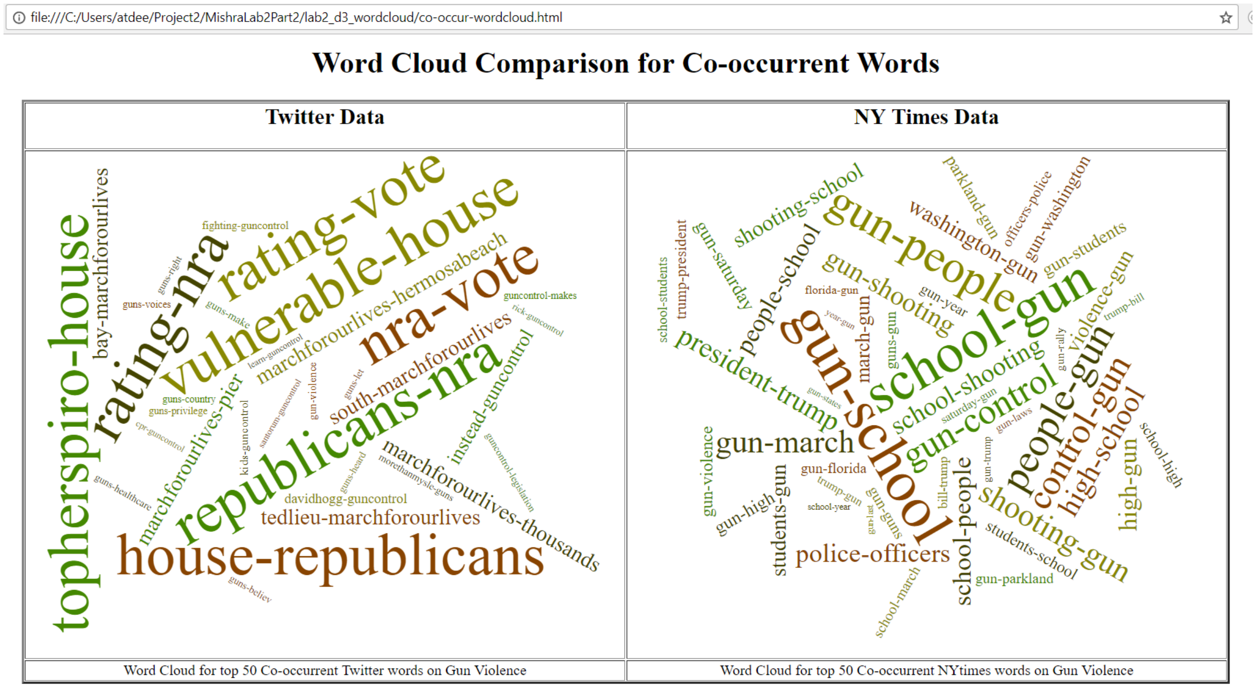
We developed a web page to show this comparison. The web page reads the word count file of Twitter and NYTimes, and sorts them on the basis of word frequency. The defined amount of words with their frequency are then fed into d3 wordcloud() function. Finally, the word cloud for top 100 words from both the sources are shown side-by-side. The web page also provides the feature to specify the word count and dynamically change the word clouds.

Additionally, we are also showing the word clouds for top co-occurrent words in Twitter and NYTimes data.

Word Cloud for Top 100 words:



Word Clouds for Top 50 Co-occurent words:



**Using word cloud library from: https://github.com/wvengen/d3-wordcloud

Commands

Starting Hadoop

```
start-hadoop.sh
```

Command to generated 'word count' and 'co-occurrence' for twitter data. Run below commands from "/lab2_hadoop/twitter/" directory:

Uploading the input file in Hadoop HDFS

```
hdfs dfs -put twt_one_day_data/
hdfs dfs -put twt_one_week_data/
```

Running the MapReduce Program for word count on twitter data for one day

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file wordcount_mapper.py -mapper wordcount_mapper.py
   -file reducer.py -reducer reducer.py -input twt_one_day_data/ -
   output twt_wrdcnt_one_dayFetching the output to the local system
```

Fetching the output to the local system

```
hdfs dfs -get twt_wrdcnt_one_day twt_wrdcnt_one_day
```

Running the MapReduce Program for word count on twitter data for week

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file wordcount_mapper.py -mapper wordcount_mapper.py
   -file reducer.py -reducer reducer.py -input twt_one_week_data/ -
   output twt_wrdcnt_one_week
```

Fetching the output to the local system

```
hdfs dfs -get twt_wrdcnt_one_week twt_wrdcnt_one_week
```

Running the MapReduce Program for co-occurrence on twitter data for one day

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file twt_cooccur_mapper.py -mapper
   twt_cooccur_mapper.py -file reducer.py -reducer reducer.py -
   input twt_one_day_data/ -output twt_cooccur_one_day
```

Fetching the output to the local system

```
hdfs dfs -get twt_cooccur_one_day twt_cooccur_one_day
```

Running the MapReduce Program for co-occurrence on twitter data for one week

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file twt_cooccur_mapper.py -mapper
   twt_cooccur_mapper.py -file reducer.py -reducer reducer.py -
   input twt_one_week_data/ -output twt_cooccur_one_week
```

Fetching the output to the local system

```
hdfs dfs -get twt_cooccur_one_week twt_cooccur_one_week
```

Command to generated 'word count' and 'co-occurrence' for NYTimes data. Run below commands from "/lab2_hadoop/ny_article/" directory:

Uploading the input file in Hadoop HDFS

```
hdfs dfs -put nyt_one_day_data/
```

```
hdfs dfs -put nyt_one_week_data/
```

Running the MapReduce Program for word count on NY Times data for one day

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file wordcount_mapper.py -mapper wordcount_mapper.py
   -file reducer.py -reducer reducer.py -input nyt_one_week_data/ -
   output nyt_wrdcnt_one_day
```

Fetching the output to the local system

```
hdfs dfs -get nyt_wrdcnt_one_day nyt_wrdcnt_one_day
```

Running the MapReduce Program for word count on NY Times data for one week

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
   2.6.4.jar -file wordcount_mapper.py -mapper wordcount_mapper.py
   -file reducer.py -reducer reducer.py -input nyt_one_week_data/ -
   output nyt_wrdcnt_one_week
```

Fetching the output to the local system

```
hdfs dfs -get nyt_wrdcnt_one_week nyt_wrdcnt_one_week
```

Running the MapReduce Program for co-occurrence on NYTimes data for one day

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -file nyt_cooccur_mapper.py -mapper nyt_cooccur_mapper.py -file reducer.py -reducer reducer.py -input nyt_one_day_data/ -output nyt_cooccur_one_day
```

Fetching the output to the local system

```
hdfs dfs -get nyt_cooccur_one_day nyt_cooccur_one_day
```

Running the MapReduce Program for co-occurrence on NYTimes data for one week

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
    2.6.4.jar -file nyt_cooccur_mapper.py -mapper
    nyt_cooccur_mapper.py -file reducer.py -reducer reducer.py -
    input nyt_one_week_data/ -output nyt_cooccur_one_week
```

Fetching the output to the local system

```
hdfs dfs -get nyt_cooccur_one_week nyt_cooccur_one_week
```

Terminating Hadoop

```
stop-hadoop.sh
```

Analysis and Results

The most frequent words in tweets:

	A	B
1	guncontrol	11683
2	marchforourlives	10749
3	nra	7799
4	gun	6931
5	gunviolence	6240
6	vote	4976
7	house	4781
8	republicans	4491
9	guns	4423
10	rating	4272
11	vulnerable	4270
12	topherspiro	4264
13	gunsense	3835
14	cpr	3502
15	march	3270

The most frequent words in articles:

	A	B
1	gun	1160
2	school	662
3	trump	556
4	people	532
5	shooting	489
6	police	482
7	year	437
8	president	372
9	high	360
10	students	315
11	march	314
12	last	312
13	officers	307
14	state	305
15	control	299

The most frequent co-occurring words in tweets (On Full data)

	A	B	C
1	nra	vote	4337
2	republicans	nra	4290
3	house	republicans	4261
4	house	vote	4261
5	rating	nra	4261
6	topherspiro	nra	4261
7	topherspiro	vote	4261
8	vulnerable	house	4261
9	vulnerable	republicans	4261
10	topherspiro	house	4260
11	topherspiro	republicans	4260
12	house	nra	4259
13	house	rating	4259
14	republicans	rating	4259

The most frequent co-occurring words in articles (On Full data)

	A	B	C
1	gun	school	3289
2	school	gun	3162
3	gun	people	2595
4	people	gun	2342
5	gun	control	2126
6	control	gun	2094
7	shooting	gun	1900
8	gun	march	1893
9	high	school	1804
10	president	trump	1733
11	gun	shooting	1697
12	school	shooting	1675
13	school	people	1672
14	police	officers	1660
15	people	school	1616

Video Recording:

Short Demo: https://buffalo.box.com/s/vhevhrfbp0uvft0hd34gsdt7v2mhs6na

Full Demo: https://buffalo.box.com/s/1nuxaenj6ug8ayv7p9qudgy62p5qu4yp

References:

D3 Word Cloud : https://github.com/wvengen/d3-wordcloud

Python Mapper and Reducer: http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/