DATA INTENSIVE COMPUTING
Lab- 2

Submitted by: Achinyta Pillai, Venkata Sai Abhishekh Sarvepalli

Immigration is one of the most important topics in the present time. World hasn't witnessed such phenomenal influx of immigrants into developed countries in the past. Presently it is one of the most debated topics by the politicians and experts. We see the social media and the rest of the internet flooded with this word. Hence, we decided to analyze the articles and tweets related to this topic. The goal of the assignment is aggregate data related to immigration from New York Times, Twitter and various other website with the help of common crawl, furthermore we will use classical big data method of MapReduce to the unstructured data collected and generate visualization.

Part 1 : Data Collection

Twitter

In order to get data from twitter we make use of its API which allows us to etch data. We aggregate data by using different keywords (such as Immigration, Legal Immigration and USCIS). The API returns us the data in JSON format. We need to make sure there are no duplicate tweets, hence we delete the retweets from the JSON data, further we iterate over max_id to get unique tweets at each iteration and only get the text and store the tweets in different text file for each of the keywords. Our text file contains more than 30,000 tweets.

New York Times

We use a similar approach to extract the articles from New York Times as we did it in twitter. We make use of the New York Times article search API which just like twitter API returns the data in a JSON format. We want to get a better understanding of the data in order to parse it; hence we have written a small function which organizes (the functions was taken from one of the blogs on how to gather the New York times data repository uploaded by one of the developers) the JSON data according to their tags. Once we have the JSON file we need to get the URL which points to the articles. Unlike twitter data we cannot get the content directly from the API data. First we get the URL and store it a csv format and delete any duplicate links which exist (in excel), then we read the links from the csv file and pass the URL into "beautifulsoup" and fetch the text under the paragraph tags (<p>) to store the data in different text files for various keywords.

Common Crawl

Common Crawl is a nonprofit organization that crawls the web and provides the contents to the public free of charge and under few restrictions. The data is hosted on Amazon S3 as part of the Amazon Public Datasets program, making it easy and affordable to scan and analyze large portions of the internet.
This format is a general archive format that can be used both for storing the raw content of a web crawl and associated metadata.
One of Common Crawl's archives contains three types of files, all in the WARC format:

- WARC files containing the raw crawl data.
- WAT files containing metadata for the WARC records in a corresponding WARC file.

- WET files containing plaintext contained in the WARC records in a corresponding WARC file.

Common crawl provides an API that allows us to query a particular index for a particular domain and returns the result in the form of URL. The data can be retrieved from the same method which we used to extract the common crawl data.

We used common crawl to get the data from various news publication and government websites namely Washington Post, CNN, BBC, News Day and United States Immigration Council (USCIS). We collected over 55,000 links related to various topics. Once we had the links we used regular expression to extract the links which are pertinent to us. We use the exact same approach to get the data from the links which we have discussed before.

Preprocessing

Before we can feed the data into our mapper and reducer we need to do a lot of preprocessing to delete the unnecessary data we get from various sources, here unnecessary means emoticons, hashtags and various other non-alphanumeric data. We accomplish this task we use regular expression to just retain the alphanumeric data and remove the rest. We run the code on the twitter data and common-crawl data.

Part 2 : Setting Up the Infrastructure

Setting up the infrastructure was one the most challenging parts in this assignment. We used multiple approaches to install Hadoop in our system and went to a number of Office Hours to meet various TAs but they were not able to help us fix the problem. Hence we went for a approach a TA provided and we installed Hadoop on our own system and ran it on our system. In the following paragraphs we would like to tell you so some of the approach's we used which was provided by the class and how it did not work.

We installed virtual machine in our system and imported the virtual image provided by the teaching assistants and followed the steps provided on piazza to setup up our NameNodes and DataNodes. However, as our code was in python we were facing multiple issues converting it into .jar files and when we went to the TA responsible for the virtual machine they were not able to help us convert our code to a .jar file.
Since we were facing this issue we downloaded the Docker and tried to work with that but the docker was throwing a variety of java exceptions and issues while trying to execute our mapper and reducer, even though our mapper and reducer were written in python. We assume the issue was that we were trying to import packages in our code, which was not doable in docker and we also had issues with python version compatibilities with the docker.

Due to all the issues I mentioned above, we finally used an idea provided by Vinooth who is one of the TAs, which was to download Hadoop on our own system and run our mapper and reducer with that.

After doing that things seem to run fine. The following are the steps used to install Hadoop on our local system, and the important commands used to run our code.

We used the information provided in the following link to help us set up Hadoop on our personal machine:
https://isaacchanghau.github.io/post/install_hadoop_mac/

After following the instruction that site we used the following list to commands to do the following tasks:

Put the Input files from our personal machine into Hadoop file system, run the mapper and reducer with python using streaming and then getting the output from the HDFS into our personal machine.

Following are the important list of commands:

// Putting "data.txt" or any file on HDFS in the following path "/user/ash/MR/input/"

1) hadoop fs -put data.txt /user/ash/MR/input/

// check if the prior command worded

2) hadoop fs -ls /user/ash/MR/input/

//This the command used to run the mapper and reducer on HDFS change the bolded //regions with your files. Output_ttm is the file I want to output myresults into.

3) hadoop jar /Users/achintyapillai/Desktop/c587/hadoop-streaming-3.1.1.jar -D dfs.blck.size=134217728 -mapper **/Users/achintyapillai/Desktop/c587/mapper.py** -reducer **/Users/achintyapillai/Desktop/c587/reducer.py** -input **/user/ash/MR/input/ttm.txt** -output **/user/ash/MR/output_ttm**

// check if the prior command worded

4) hadoop fs -ls /user/ash/MR/output_ttm/

// Transfer the output file to your own directory (The bolded part)

5) hadoop fs -get /user/ash/MR/output_ttm/ **/Users/achintyapillai/Desktop/c587/**

// cd to the output file location in local directory and run the following for top 10 words

6) cat part-00000.txt | sort -n -k2 | tail –n10

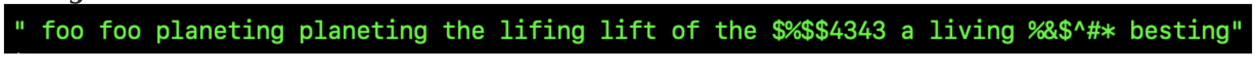
Part 3: Processing and results

For this part I would like to show how we have actually processed our data and run it on our map reduce.

The preprocessing of data for twitter, New York Times and Common Crawl was all done using my mapper.py and reducer.py files for the word count and using the comap.py for the word co-occurrence. By pre-processing I mean, cleaning the text to only get text in the English language, removing emoticons, trimming lines, stemming and stopping words. Removing repeated tweets was removed using our python code.

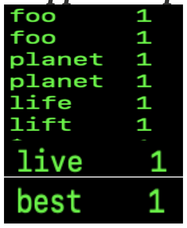
After collecting and processing our data we were finally able to run it through our mapper for word count first. The mappers task was to identify the individual words and relate it to a count of 1. The output for the mapper given some text, for example looked like the following:

Text given:



```
" foo foo planeting planeting the lifing lift of the $%$$4343 a living %&$^#* besting"
```

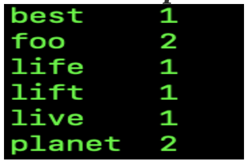
Mapper Output:



```
foo      1
foo      1
planet   1
planet   1
life     1
lift     1
live     1
best     1
```

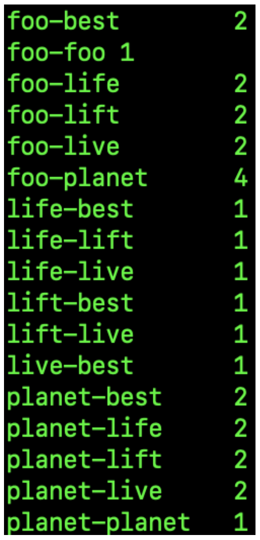
The reducers task was to count the number of times one individual instance of word found by the mapper and relate it to the total count. We used the git hub repositories mentioned by teaching assistants to help understand MR.
The output for that is given below:

Reducer output:



```
best      1
foo       2
life      1
lift      1
live      1
planet    2
```

For the Word co-occurrence for the same sentence the final output was like the following ***(Use comap.py as the mapper for Word co-occurrence and reducer1.py as the reducer for Word co-occurrence):***

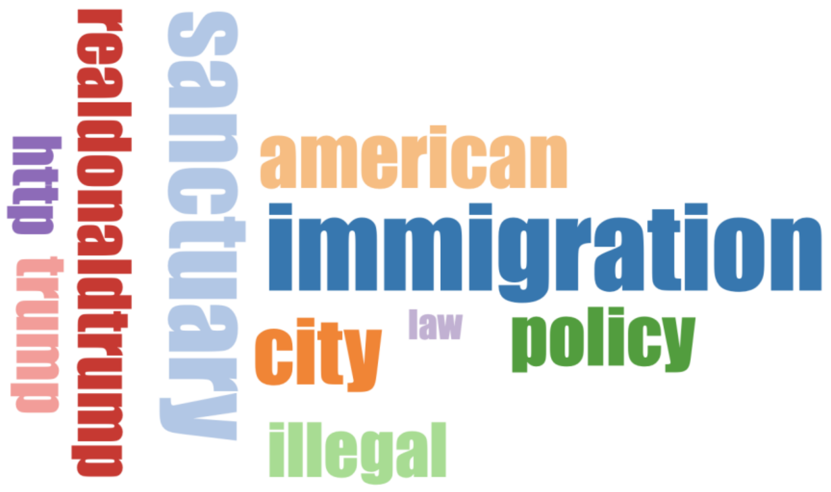


```
foo-best        2
foo-foo 1
foo-life        2
foo-lift        2
foo-live        2
foo-planet      4
life-best       1
life-lift       1
life-live       1
lift-best       1
lift-live       1
live-best       1
planet-best     2
planet-life     2
planet-lift     2
planet-live     2
planet-planet   1
```

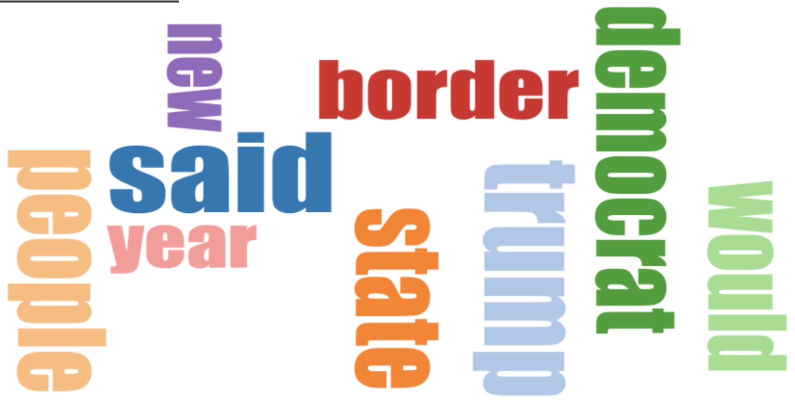
Word Count for Twitter, New York times and Common Crawl:

For the twitter data, New York Times data and Common Crawl after getting the word count results from reducer, using HTML and D3.js I have made a word could for the representation (I have normalized the count for the top ten words as 5000 count is too big to show on a screen for sizing purposes) for the results. The followings are examples of some of the word clouds (sometime in the word could due to sizing issues all the words might not show up, if 10 words don't show up just refresh the until 10 show up) I have made for twitter, New York times and common crawl:

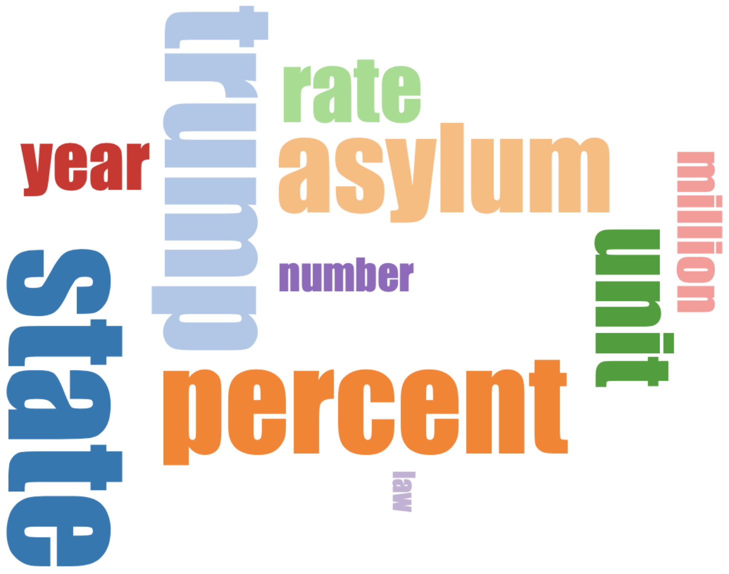
Twitter:



New York Times:

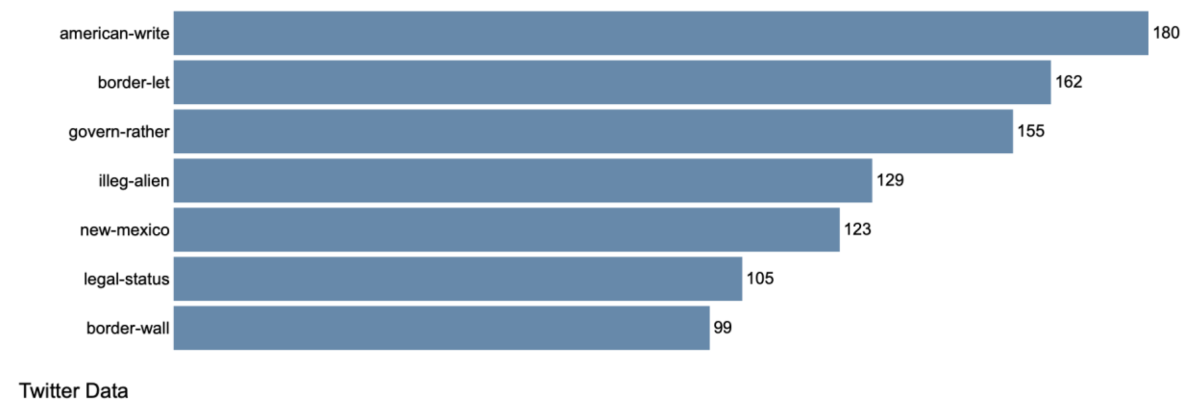


Common Crawl:

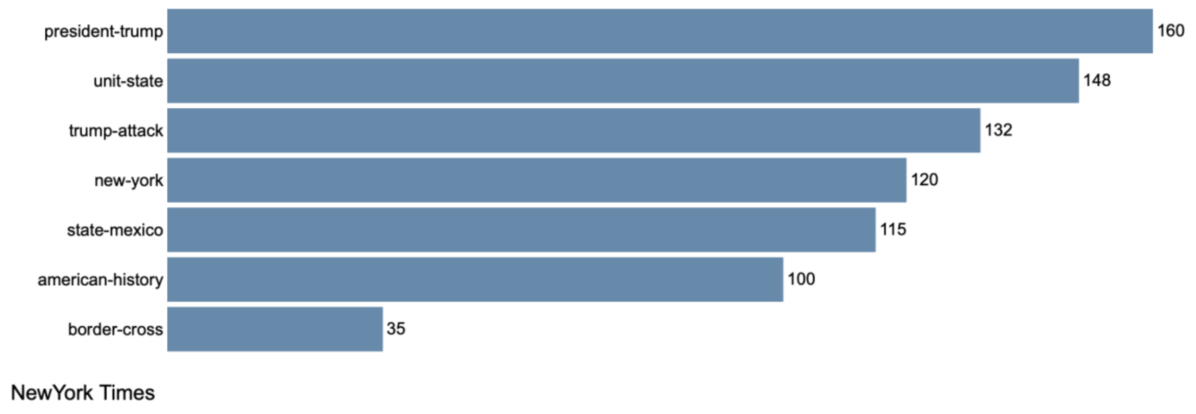


For the word co-occurrence, I have used a bar graph for visualization and representation and it is as follows:

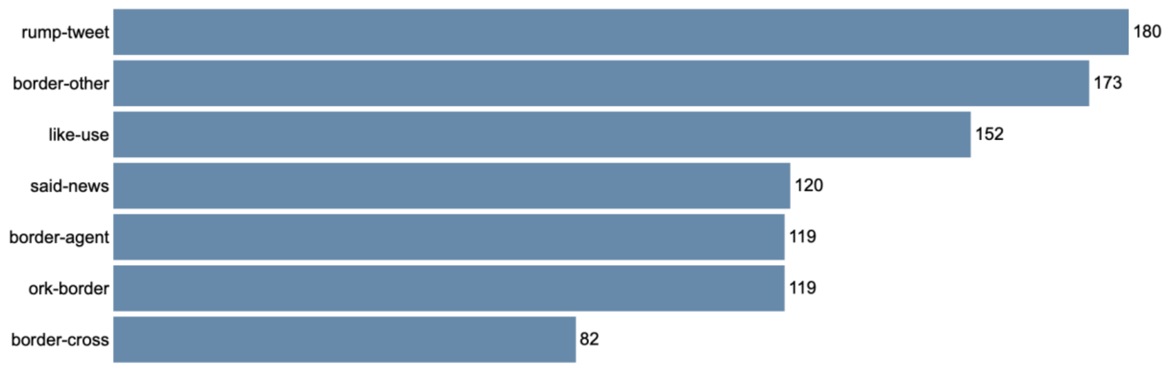
Twitter:



New York Times:

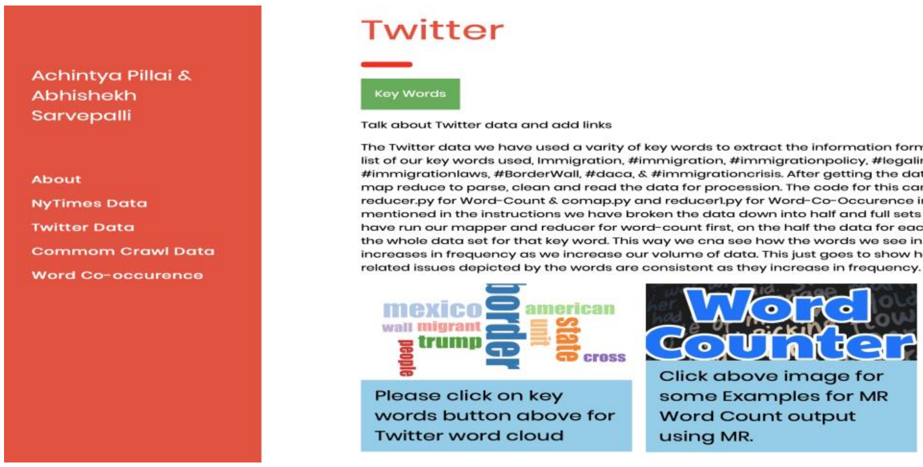


Common Crawl:



Common Crawl

Web-Site:



Twitter

Talk about Twitter data and add links

The Twitter data we have used a varity of key words to extract the information form various articles. The following is a list of our key words used, Immigration, #immigration, #immigrationpolicy, #legalimmigration, #illegalimmigration, #immigrationlaws, #BorderWall, #daca, & #immigrationcrisis. After getting the data using the words, we have used map reduce to parse, clean and read the data for procession. The code for this can be found in mapper.py and reducer.py for Word-Count & comap.py and reducer1.py for Word-Co-Occurence in our submission folder. As mentioned in the instructions we have broken the data down into half and full sets and done the word processing. We have run our mapper and reducer for word-count first, on the half the data for each key word, then second we ran it for the whole data set for that key word. This way we cna see how the words we see in the half dataset word cloud increases in frequency as we increase our volume of data. This just goes to show how our hypothesis that the the related issues depicted by the words are consistent as they increase in frequency.

Please click on key words button above for Twitter word cloud

Click above image for some Examples for MR Word Count output using MR.

The web-site is an interactive way to look at our visualizations for the results for of MR jobs. On the left hand side of the screen you will notice a tabs related to New York Times data, twitter data, common crawl data and word co-occurrence for all three. Please navigate to each of the tabs and click on key words (green button) and select the visualization you would like to see.

To go to the site: go to my submission folder/website/index.html
Or file:///Users/achintyapillai/Desktop/c587/website/index.html#showcase
Change the highlighted part to your directory and part it as a url on chrome or safari and run.

References

1. https://www.bellingcat.com/resources/2015/08/13/using-python-to-mine-common-crawl/
2. https://dlab.berkeley.edu/blog/scraping-new-york-times-articles-python-tutorial