Big-Data Content Retrieval, Storage, and Analysis

(Crisis in Ukraine)

By: Amit Kamat
University at Buffalo

1. Abstract

The Big-Data Content Retrieval, Storage, and Analysis project gives one a fantastic exposure to data wrangling, map-reduce, and web-enabled data visualization. Through the use of various tools we select a trending topic, in this case the crisis in Ukraine, and collect relevant data that we subsequently explore.

I plan on implementing this project in three parts. First and foremost I need a data wrangling technique for which I will use python, Tweepy, and the Twitter API. I will refactor the data into a predefined format and make sure to leave only relevant information within, removing stop words, links, and other unnecessary parts. I will then proceed onto the map reduce portion of the project where I shall implement the word count, co-occurrence, k-means, and shortest path algorithms for the data. I plan on using the in-mapper combiner technique wherever possible. Both pairs and stripes approach, including relative frequency, will be implemented for the co-occurrence. For the shortest path problem I plan to use replies, retweets, and mentions, basically anything starting with a '@' in a tweet, as the connection between nodes. Last but not least I will implement the visualizations using Gephi coupled with Sigmajs Exporter.

As a creative venture I plan on collecting origin coordinates for tweets while utilizing Tweepy and the Twitter API. I will use those to create a nice plot on the world map. I personally think it will be an interesting thing to see.

2. Project Objectives

The Big-Data Content Retrieval, Storage, and Analysis project, at the very least, will meet the following list of objectives:

- Learn about data wrangling by utilizing the Twitter API.
- Learn more about big data collection and analysis.
- Learn to use Hadoop and map-reduce.
- Learn more about distributed file systems such as HDFS.
- Explore different techniques and algorithms relevant to map-reduce.
- Explore different techniques in web-enabled data visualization.
- Provide useful metrics and visualizations.

3. Project Approach

First and foremost I plan on reading (and re-reading) Chapters 3 and 4 of *Data-Intensive Text Processing with MapReduce* until I feel comfortable with the algorithms presented there. Once I am ready to proceed I will commence with deciding on a data wrangling technique. I am thinking of using Tweepy because I have had it as a goal to get better at Python. I plan on cleaning up the data by removing stop words, links and other unneeded elements. I also plan on delimiting the data in such a manner so that different elements can be easily distinguished and retrieved. I will have python scripts intended to combine/split the collected data appropriately. I will definitely collect more than 200000 tweets (363000 to be exact).

Once I am done with the wrangling, I will set up hadoop and start along. I plan of doing as much as possible directly using hadoop map-reduce in order to get good at it. This is why I will do the word count problem using an in-mapper combiner. This is also, why I plans to use secondary sort in Hadoop when it is generally perfectly fine to do the sort for most trending words using other methods.

The co-occurrence problem has two parts, pairs and stripes. Both will have in-mapper combiners. I will write my own Writable Comparable for the pairs, which should be interesting. I will also have a petitioner. For both the algorithms, I will follow the methodology described in the book as closely as possible. Just for fun I will have a sorted by count map reduce function in place.

For K-means, I will read the post provided in the project description. Once I understand it I shall commence. I plan to use counters for this algorithm to pass along data between iterations. Once the centroids are found, I will create files listing users, their followers, and respective groups. For more visualizations, I will run word count and co-occurrence for each group separately as well.

Finally yet importantly in terms of algorithms I will write out the shortest path. I plan to follow what is described in the book as closely as possible. I will use my most followed user as the start of the graph. I will initialize the graph itself by finding points that are somewhat midpoints in the set of follower counts. I will then run the regular algorithm, which should provide me with a nice labeled graph.

For visualizations, I will use Gephi and sigma.js. Since the data set is very large, it will be very nice to look at it through a web-enabled visualization. Using Python I will create scripts, which reformat the regular outputs into graph files for Gephi. I will then export those graphs using Sigmajs Exporter.

4. Data Format and Source

The data source is through the twitter API, which is set to stream tweets containing information related to Ukraine, Russia, and the ongoing crisis there. I am specifically using Tweepy to collect the data. I will refactor the data in such a manner as to leave only relevant information, removing stop words, links, and other irrelevant details. I have provided a README.txt in the /data/ folder where you can find more information about the format and source. Here is a copy for your reference:

```
==================================================================================
This folder contains the following:
==================================================================================
input/ - folder contains the input data used for map reduce through this project
        363000 tweets in total
output/ - the contents of this folder are the map reduce output for this project

==================================================================================
Input details:
==================================================================================
Files:
----------
input/data_1.txt - Ukraine crisis data acquired from twitter with an aggregator
input/data_2.txt   these files contain 363000 tweets. 1 being the oldest, and 3
input/data_3.txt   the newest.

Format:
----------
Username | Followers_count | Mentions | Hashtags | Tweet

==================================================================================
Wordcount output details:
==================================================================================
Files:
----------
output/wordcount/part-r-00000 - unsorted wordcount output hashtags, words, & @
output/wordcount/part-r-00001
output/wordcount/sorted/part-r-00000 - sorted wordcount output full
output/wordcount/sorted/words-r-00000 - sorted wordcount output words only
output/wordcount/sorted/hashtags-r-00000 - sorted wordcount output hashtags only
output/wordcount/sorted/atsign-r-00000 - sorted wordcount output @ only

Format:
----------
word,count
```

==
Co-occurrence output details:
==
Files:
----
output/coocurrence/pairs/part-r-00000 - alphabetical pairs output
output/coocurrence/pairs/part-r-00001
output/coocurrence/stripes/part-r-00000 - alphabetical stripes output
output/coocurrence/stripes/part-r-00001
output/coocurrence/sorted/part-r-00000 - sorted by count co-occurrence output
output/coocurrence/pairs_five/part-r-00000 - alphabetical pairs with 5 reducers
output/coocurrence/pairs_five/part-r-00001
output/coocurrence/pairs_five/part-r-00002
output/coocurrence/pairs_five/part-r-00003
output/coocurrence/pairs_five/part-r-00004

Format:
----
hashtag_1,hashtag_2,count,relative_frequency

==
K-means output details:
==
Files:
----
output/centroids.csv - contains K-means iterations begining with the most recent

Format:
----
iteration_#,cluster_id(low),cluster_coordinates(low),rows_count(low),
cluster_id(medium),cluster_coordinates(medium),rows_count(medium),
cluster_id(high),cluster_coordinates(high),rows_count(high)

Files:
----
output/kmeans/low-r-00000 - input/data.txt split for the low cluster
output/kmeans/medium-r-00000 - input/data.txt split for the medium cluster
output/kmeans/high-r-00000 - input/data.txt split for the high cluster

Format:

Username | Followers_count | Mentions | Hashtags | Tweet

Files:
----
output/kmeans/part-r-00000 - sorted list of all users by cluster
output/kmeans/userslow-r-00000 - sorted list of users in the low cluster
output/kmeans/usersmedium-r-00000 - sorted list of users in the medium cluster
output/kmeans/usershigh-r-00000 - sorted list of users in the high cluster

Format:
----
username,cluster_id,followers_count

```
--------------------------------------------------------------------------------
Files:
----------
output/kmeans/low/ - wordcount & co-occurrence output for the low cluster
output/kmeans/medium/ - wordcount & co-occurrence output for the medium cluster
output/kmeans//high/ - wordcount & co-occurrence output for the high cluster

Format:
----------
Please refer to wordcount and/or co-occurrence output details.

================================================================================
Shortest path output details:
================================================================================
Files:
----------
output/spath/part-r-00000 - graph data shortest path network for cnnbrk (CNN
                    breaking news) on twitter. Connections are mentions,
                    replies, and retweets bearing a certain username.

Format:
----------
username,distance,connection_1:conntection_2:...:connection_n:
```

5. Amount of data collected and details

I have collected **363000 tweets** spanning a period from March 27th to April 7th. The data was not collected continuously rather for different sets of hours every day. The data is collected in the predefined format seen in section 4 that is for each line in the file(s) we have the following:

Username | Followers_Count | Mentions | Hashtags | Tweet

This data is intended to provide us with details on the trends and news relating to the crisis in Ukraine. Using it we will be able to find how the data travels around twitter, and which sort of words, hash tags, and users are most influential.

6. Aggregator details (from scratch)

My aggregator is built from scratch using Tweepy. The reason for this is that I wanted to get better at Python by using it in this assignment. The aggregator not only collects data but also removes unnecessary elements and formats it in the notation seen above. Some of the elements I clean from the data are stop words, links, symbols and numbers (except for #), and others such. I format the data in a nice, clean, and easy to access manner so that my life would be much simpler through the map-reduce phase of the assignment. Along with the aggregator, I also provide some optional Python scripts to combine/split the data files. Here are the details, which can also be found in the README.txt within the /aggregator/ directory:

```
-------------------------------------------------------------------------------
This folder contains the following files:
-------------------------------------------------------------------------------
aggregator.py - aggregator for twitter data for the the situation in Ukraine
refactor.py   - combines and refactors the data files
split.py      - splits the combined file from refactor.py into desired amount of
                chunks

-------------------------------------------------------------------------------
To run the aggregator you need to have the following installed:
-------------------------------------------------------------------------------
Python 2.7.x
Tweepy
nltk.corpus (including stopwords)
```

```
------------------------------------------------------------------------------
To install the above you need to do the following:
------------------------------------------------------------------------------
1. Download and install Python 2.7.x from: https://www.python.org/downloads
2. Download and install Setuptools from: http://pypi.python.org/pypi/setuptools
3. Install Pip by running: sudo easy_install pip
4. Install NLTK by running: sudo pip install -U pyyaml nltk
5. Install Tweepy by running: sudo pip install tweepy
6. Run: python -c 'import nltk; nltk.download()'
7. Select the 'Corpora' Tab browse down to stopwords and click on 'Download'

------------------------------------------------------------------------------
To run the aggregator edit aggregator.py as follows:
------------------------------------------------------------------------------
- Add your Twitter API key on line 13
- Add your Twitter API secret on line 14
- Add your Twitter access token on line 17
- Add your Twitter access token secret on line 18
- (Optional) If you need to change the topic edit line 28

------------------------------------------------------------------------------
Now that you are all set up you can run the aggregator:
------------------------------------------------------------------------------
python aggregator.py

------------------------------------------------------------------------------
(Optional) When you are done wrangling refactor the data by running :
------------------------------------------------------------------------------
python refactor.py

------------------------------------------------------------------------------
(Optional) If you want to split the combined file from refactor run :
------------------------------------------------------------------------------
python split.py number_of_splits

------------------------------------------------------------------------------
You will now have the following output files:
------------------------------------------------------------------------------
data/ - contains data files in chunks of 1000 tweets each
data.txt - all the tweets in data/ combined into one big data file
coordinates.csv - geocodes used for tweet origin visualizations
```

Aggregator (aggregator.py) details:

The aggregator is set up to remove the following elements from a raw tweet:
- Symbols (except for '#' in hashtags)
- Links
- Retweet characters ('rt @')
- Numbers
- Stopwords (from NTLK Corpus containing over 2400 stopwords)

The aggregator collects the following information:
- Username
- Followers count
- Mentions ('@' or 'rt @')
- Tweet
- Hashtags
- Tweet coordinates (in a separate file due to low frequency)

The aggregator collects data in the folder './data'. The data is contained in files using the date and time of their creation as a name. Each of those files contains a 1000 tweets.

Refactor (refactor.py) details:

Refactor.py goes through all the files in './data' and removes those which are deemed incomplete (having less than a 1000 tweets). It then combines all the tweets from each file into a single large data file './data.txt'.

Split (split.py) details:

Once refactor.py is done running you will have a './data.txt' combined file. You can split this file by running:

```
python split.py number_of_splits
```

Data format is as follows per line:

Username | Followers_count | Mentions | Hashtags | Tweet

Example:
dmytrokovalov | 13 | usosce | #crimea #ukraine #unitedforukraine | mistake |

Example when a filed is missing (Mentions / Hashtags / Tweet):
talkvietnambuzz | 538 | | | rejects russian annexation crimea |

Coordinates format:

The coordinates are collected in 'coordinates.csv' in two columns the first with longitudes and the second with latitudes. This data is intended to be used separately due to being rarely provided along with the tweet. It is collected as a creative venture and will not be used during the MR phase of this project.

<u>7. MR (mapper, reducer pseudo code)</u>

7.1 – Word Count

```
class Mapper
    H <- new HashMap<term, count>
    method map(position p, line l)
        words <- hashtags, mentions, and tweet in line l
        for all term t in words do
            sum <- if H.get(t) is null then 1 else H.get(t) + 1
            H.put(t, sum)
    method cleanup
        for all term t in H do
            Emit(term t, count H.get(t))
class Reducer
    method reduce(term t, counts [c_1,c_2,…])
        sum <- 0
        for all count c in counts [c_1,c_2,…] do
            sum <- sum + c
        Emit(term t, count sum)
```

7.2.1 – Co-Occurrence (Pairs)

```
class pair
    term a
    term b
    method getFirst
        return min(a, b)
    method getSecond
        return max(a, b)
    method compare(pair p)
        if this.getFirst equals p.getFirst
            if p.getSecond is *
                return more
            if this.getSecon is *
                return less
            if this.getSecond > p.getSecond
                return more
            else if this.getSecond < p.getSecond
                return less
            else
                return equals
        else
            if this.getFirst < p.getFirst
                return less
            else
```

```
                        return more
class Mapper
    H <- new HashMap<pair, count>
    method map(position p, line l)
        words <- hashtags in line l
        S <- new SortedSet<term>
        for all term t in words do
            S.add(term)
        while S.size > 1
            hashtag <- S.first()
            marginal <- new pair(hashtag, *)
            mSum <- if H.get(marginal) is null then 0 else H.get(marginal)
            H.remove(hashtag)
            For all term neighbor in S do
                mSum <- mSum + 1
                p <- new pair(hashtag, neighbor)
                sum <- if H.get(p) is null then 1 else H.get(p) + 1
                H.put(p, sum)
            H.put(marginal, sum)
    method cleanup
        for all pair p in H do
            Emit(pair p, count H.get(p))
class Partitioner
    getParitioner(pair p, count c, count numParitions)
        return abs(p.first().hashCode()) % numParitions
class Reducer
    first <- new term
    mSum <- 0
    method reduce(pair p, counts [c_1,c_2,…])
        sum <- 0
        if p.getFirst not equals first
            first <- pair.getFirst
            mSum <- 0
            for all count c in counts [c_1,c_2,…] do
                mSum <- mSum + c
        else if p.getSecon equals *
            for all count c in counts [c_1,c_2,…] do
                mSum <- mSum + c
        else
            for all count c in counts [c_1,c_2,…] do
                sum <- sum + c
            r <- new CountRelative(sum, sum / marginalSum)
            Emit(pair p, CountRelative r)
```

7.2.2 – Co-Occurrence (Stripes)

```
class Mapper
    H <- new HashMap<term, HashMap<term, count>>
    method map(position p, line l)
        words <- hashtags in line l
        S <- new SortedSet<term>
        sum <- 0
        for all term t in words do
            S.add(term)
        while S.size > 1
            hashtag <- S.first()
            stripe <- null
            for all term t in words do
                if stripe is null
                    stripe <- new new HashMap<term, count>
                sum <- if stripe.get(t) is null then 1 else stripe.get(t) + 1
                stripe.put(t, sum)
            H.put(hashtag, stripe)
    method cleanup
        for all term t in H do
            stripe <- H.get(t)
            Emit(term t, HashMap<term, count> stripe)
class Reducer
    method reduce(term t, HashMap<term, count> stripes [s_1,s_2,…])
        stripe <- new TreeMap<term, count>
        mSum <- 0
        for all element in stripes [s_1,s_2,…] do
            for all term t in element do
                stripe.put(t, element.get(t))
                mSum <- mSum + element.get(t)
        while stripe.size() is not 0
            key <- stripe.firstKey()
            sum <- stripes.get(key)
            pair <- new pair(t, key)
            r <- new CountRelative(sum, sum / mSum)
            Emit(pair, r)
            Stripe.remove(key)
```

7.3 – K-means

```
class Mapper
    centroids <- null
    H <- new HashSet<term>
    totalRecords <- new GroupArray<count> {LOW, MEDIUM, HIGH}
    totalCoordinates <- new GroupArray<count> {LOW, MEDIUM, HIGH }
```

```
        method map(position p, line l, context c)
                if centroids is null
                        centroids <- getCentroids()
                        c.setIteration(centroids.getIteration())
                username <- username in line l
                followersCount <- followersCount in line l
                if not H.contains(username)
                        H.add(username)
                        group <- getGroup(followers)
                        totalRecords[group] <- totalRecords[group] + 1
                        totalCoordinates[group] <- totalCoordinates[group] + 1
        method cleanup
                if TotalRecords[LOW] is not 0
                        r <- new RecordAverage(
                                totalRecords[LOW],
                                totalCoordinates[LOW]/totalRecords[LOW])
                        Emit(LOW, r)
                if TotalRecords[MEDIUM] is not 0
                        r <- new RecordAverage(
                                totalRecords[MEDIUM],
                                totalCoordinates[MEDIUM]/totalRecords[MEDIUM])
                        Emit(MEDIUM, r)
                if TotalRecords[HIGH] is not 0
                        r <- new RecordAverage(
                                totalRecords[HIGH],
                                totalCoordinates[HIGH]/totalRecords[HIGH])
                        Emit(HIGH, r)
Class Reducer
        method reduce(term type, RecordAverage ras [r_1,r_2,…], context c)
                totals <- 0
                records <- 0
                for all ra in ras do
                        total = total + ra.getAverage() * ra.getRecord()
                        records = records + ra.getRecord()
                if type is LOW
                        context.setLow(totals / records)
                        context.setCountLow(records)
                else if type is MEDIUM
                        context.setMedium(totals / records)
                        context.setCountMedium(records)
                else
                        context.setHigh(totals / records)
                        context.setCountHigh(records)
        method cleanup(context c)
                c.incrementIteration()
```

7.4 – Shortest Path

```
class Mapper
    method map(position p, line l)
        username <- username in line l
        distance <- distance in line l
        adjacency <- adjacency in line l
        Emit(username, <adjacency, distance>)
        for all node in adjacency do
            Emit(username, distance + 1)
Class Reducer
    sortedByDistance <- new TreeMap<count, TreeMap<term,  TreeSet<term>>>
    method reduce(term key, NodeOrDistance values [nod_1,nod_2,...], context c)
        min <- INFINITY
        node <- new Node()
        sortedByKey <- null
        for all value in values do
            if value.isNode()
                min <- node.getDistance()
        else if value.getDistance() < min
            min <- value.getDistance()
        node.setDistance(min)
        if min equals INFINITY
            context.IncrementInfinity()
        if context.INFINITY equals 0
            sortedByKey <- sortedByDistance.get(min)
            if sortedByKey is null
                sortedByKey <- new TreeMap<term,  TreeSet<term>>
            sortedByKey.put(key, node)
            sortedByDistance.put(min, sortedByKey)
        else
            Emit(key, node)
    method cleanup
        while sortedByDistance is not empty do
            sortedByKey <- sortedByDistance.getFirstValue
            while sortedByKey is not empty do
                key <- sortedByKey.firstKey()
                value <- <sortedByKey.firstValue,
                          sortedByDistance.firstKey>
                Emit(key, value)
                sortedByKey.removeFirst()
            sortedByDistance.removeFirst()
```

8. Configuration of the cluster used

The cluster was run locally on Mac OS X using Hadoop, HDFS, and Yarn. It was tested with different numbers of mappers and reducers for different algorithms. Here is the general configuration and installation/running instructions provided along with the README.txt in the /hadoop_project/ directory:

```
--------------------------------------------------------------------------------
This folder contains the following files:
--------------------------------------------------------------------------------
project2.tar - bundle containing my MR project for hadoop
project2.jar - a jar file of the above project

--------------------------------------------------------------------------------
To run this project you need the have the following installed and configured:
--------------------------------------------------------------------------------
Java 1.6
Hadoop 2.2.0
Eclipse Kepler

--------------------------------------------------------------------------------
Tested on the following environment:
--------------------------------------------------------------------------------
OS X 10.8.5
Java 1.6.0_65 (http://support.apple.com/kb/DL1573)
Hadoop 2.2.0

Setup using the following guide:
hadoop_install_with_eclipse_plugin_and_code_changes.docx
https://piazza.com/class_profile/get_resource/hog616iewln1p/hsf9hbalzal3f9

Hadoop installation directory:
/Applications/hadoop-2.2.0

On Mac OS X you may want to run the following command if the jar isn't working:
sudo ln -s /usr/bin/java /bin/java

--------------------------------------------------------------------------------
(Optional) Edit the source code to reflect for your own environment:
--------------------------------------------------------------------------------
You can change the default input path changing the DEFAULT_INPUT variable in
'src/common/Common.java'.  This variable defaults to the "/input" directory.

In order tor run this project directly from Eclipse you may want to change lines
63 and 64 in 'src/common/Common.java' to contain the path to your core-site.xml
and hdfs-site.xml files in your hadoop deployment directory.
```

```
------------------------------------------------------------------------------
Initialize the hadoop:
------------------------------------------------------------------------------
1. Run: 'start-all.sh'

Alternatively run:
1. start-dfs.sh
2. start-yarn.sh

------------------------------------------------------------------------------
DFS locations setup:
------------------------------------------------------------------------------
The default input directory which is going to be used is "/input".

Please perform the following steps:
1. Run: hadoop fs -mkdir /input
2. Run: hadoop fs -put <location_of_input> /input

You can use the input file provided (363000 tweets) in data/input/data.txt

------------------------------------------------------------------------------
Input format:
------------------------------------------------------------------------------
The only input format you will use is the one specified by the aggregator. All
data needed for the project parts can be contained and/or derived from it.

Format:
----------
Username | Followers_count | Mentions | Hashtags | Tweet

------------------------------------------------------------------------------
Import the project in Eclipse:
------------------------------------------------------------------------------
1. In Eclipse go to File -> Import -> General -> Existing Project into Workspace
2. Browse and select project2.tar in the 'Select archive file' field
3. Click on Finish

------------------------------------------------------------------------------
To run the project directly from Eclipse:
------------------------------------------------------------------------------
1. Set up the hadoop plugin for eclipse:
   https://github.com/winghc/hadoop2x-eclipse-plugin
2. You may use the above mentioned guide for configurations:
   https://piazza.com/class_profile/get_resource/hog616iewln1p/hsf9hbalzal3f9
3. Set up the optional step above by setting lines 63 and 64 for 'Common.java'
4. Set the following VM arguments in Eclipse: -Xms512M -Xmx1524M
   They need to be set in the project's Run Configurations -> Arguments
5. To run everything run: 'src/driver/All.java'
6. For wordcount run: 'src/driver/WordCount.java'
7. For co-occurrence run: 'src/driver/CoOccurrence.java'
8. For pairs only run: 'src/driver/Pairs.java'
9. For stripes only run: 'src/driver/Stripes.java'
10. For k-means with addition outputs run: 'src/driver/Kmeans.java'
11. For k-means only run: 'src/driver/KmeansOnly.java'
12. For shortest path run: 'src/driver/ShortestPath.java'
```

To run the project using a jar file:
----
1. Set the following VM arguments in Eclipse: -Xms512M -Xmx1524M
   They need to be set in the project's Run Configurations -> Arguments
2. In eclipse go to File -> Export -> Java -> JAR file
3. Select project2
4. Make sure 'Export generated class files and resources' is selected
5. In the Jar file: filed enter <desired_location>/project2.jar
6. Click on Finish
7. With your terminal navigate to: cd <desired_location>
8. To run everything run: hadoop jar project2.jar driver.All
9. For wordcount run: hadoop jar project2.jar driver.WordCount
10. For co-occurrence run: hadoop jar project2.jar driver.CoOccurrence
11. For pairs only run: hadoop jar project2.jar driver.Pairs
12. For pairs with 5 reducers run: hadoop jar project2.jar driver.PairsFive
13. For stripes run: hadoop jar project2.jar driver.Stripes
14. For k-means with addition outputs run: hadoop jar project2.jar driver.Kmeans
15. For k-means only run: hadoop jar project2.jar driver.KmeansOnly
16. For shortest path run: hadoop jar project2.jar driver.ShortestPath

----
(Optional) Additional arguments:

It is possible to specify input arguments for the tests.
args[0] -> Input path (default is "/input")
args[1] -> Output path (default is "/output")

<u>9. Outputs: different outputs, and visualizations</u>

You can learn more about the textual output in section 4 (Data Format and Source). I will provide some snippets of data here, however I suggest looking in the /data/ directory for more details.

I will also provide some screenshots of visualizations. This screenshots are not intended to provide complete details since the output is web based. You can look at it in more details in Safari / Firefox (does not work with Chrome for some reason).

The visualizations are found in the /visualizations/sigmajs/ folder. You can also import the gdf files in /visualizations/gdf in Gephi and look at them in more detail there as well.

9.1 - Word Count

Top 10 All:
```
russia,123059
@russia,117943
ukraine,100566
russian,81547
@russian,80954
@ukraine,76711
putin,55084
@putin,53923
#ukraine,40475
crimea,28838
```

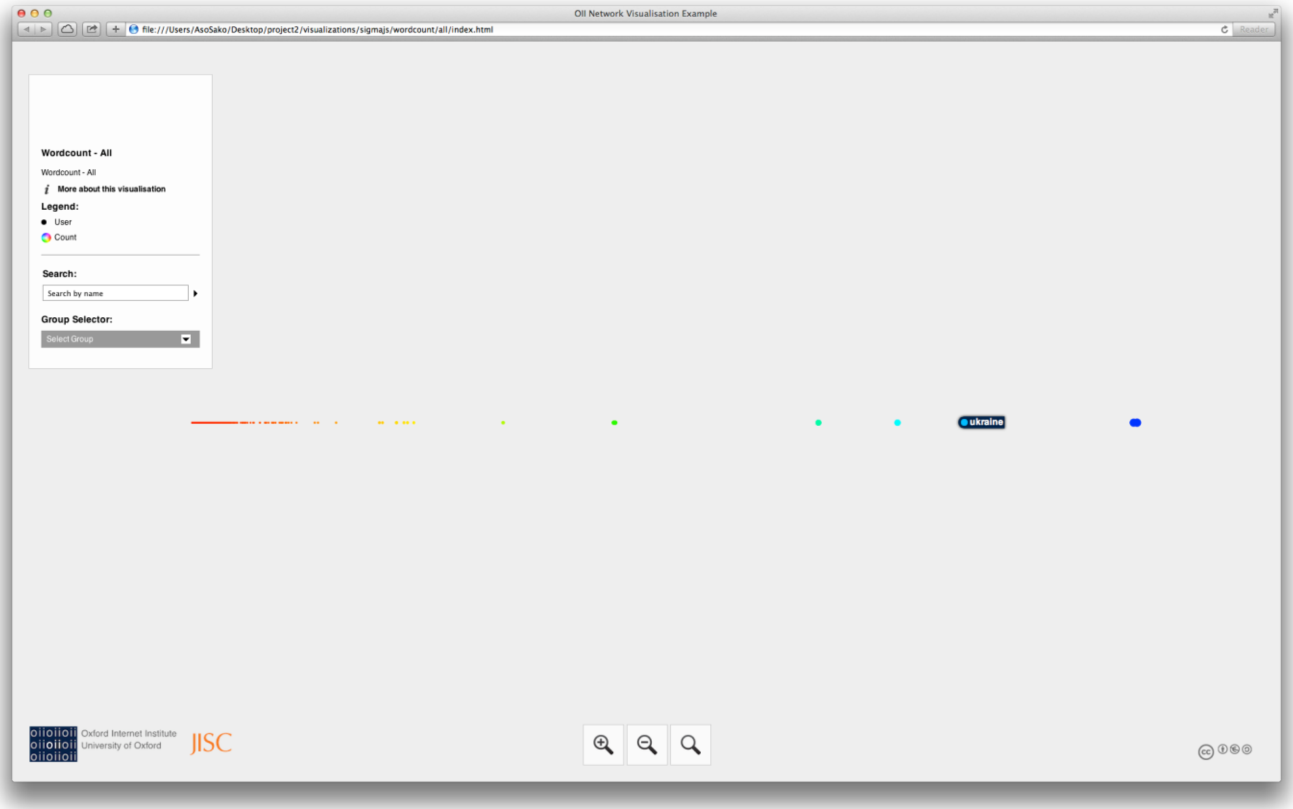
Top 10 Words:
```
russia,123059
ukraine,100566
russian,81547
putin,55084
crimea,28838
obama,28017
nato,26602
crisis,18719
amp,16355
via,15934
```

Top 10 Hash tags:
```
#ukraine,40475
#russia,24761
#crimea,12894
#putin,6468
#russian,5377
```

#news,3573
#tcot,3501
#nato,2696
#us,2274
#yalta,2261

Top 10 Mentions:
@russia,117943
@russian,80954
@ukraine,76711
@putin,53923
@nato,26536
@obama,20391
@news,11279
@military,8929
@crimea,8715
@president,8373

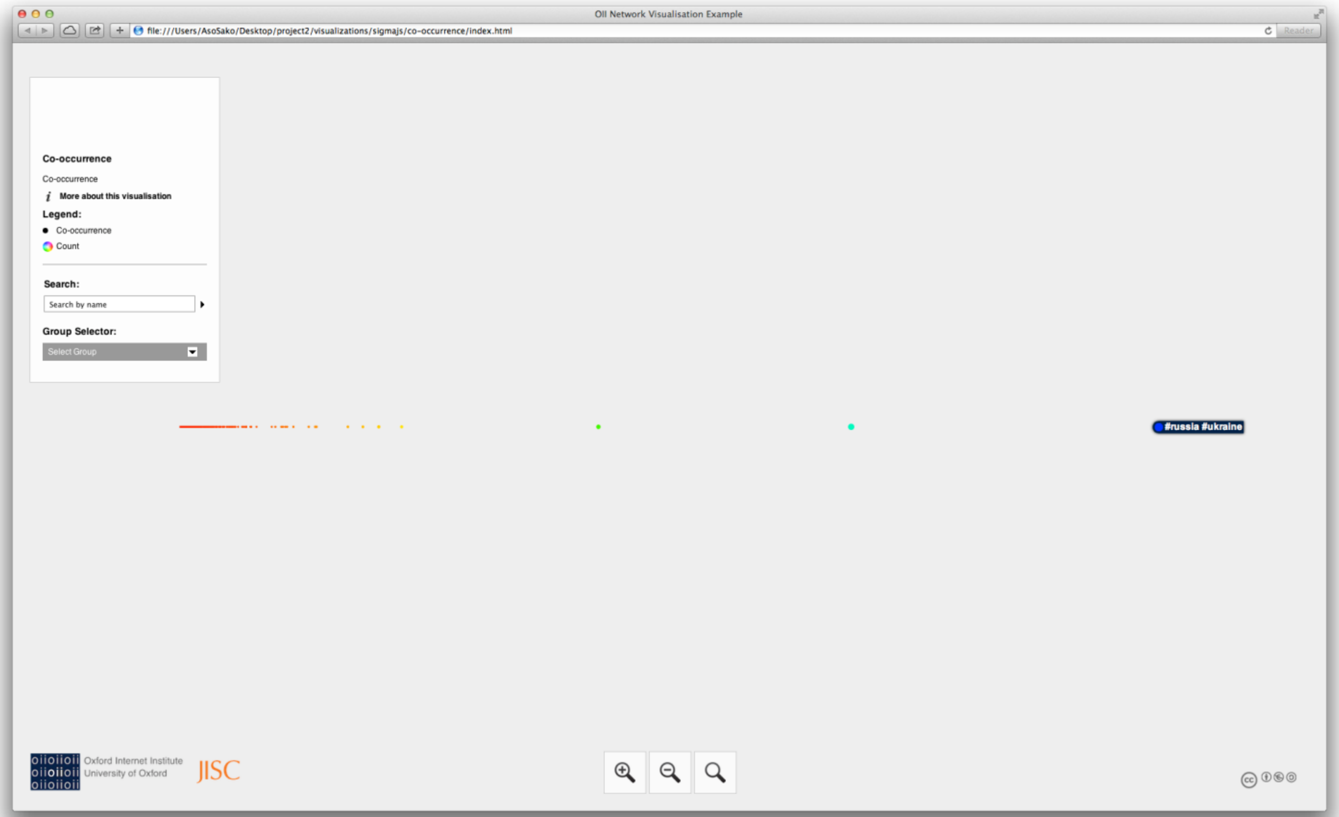
General Look and Feel of Word Count Visualizations:


OII Network Visualisation Example
file:///Users/AsoSako/Desktop/project2/visualizations/sigmajs/wordcount/all/index.html
Wordcount - All
Wordcount - All
More about this visualisation
Legend:
● User
Count
Search:
Search by name
Group Selector:
Select Group
ukraine
Oxford Internet Institute
University of Oxford
JISC

9.2 - Co-Occurrence

Top 10 (Counts):
#russia,#ukraine,11109,0.49203
#crimea,#ukraine,7618,0.27591
#crimea,#russia,4748,0.17197
#putin,#ukraine,2514,0.30995
#crimea,#yalta,2256,0.08171
#ukraine,#yalta,2253,0.2887
#russian,#ukraine,2074,0.44222
#putin,#russia,1903,0.23462
#russia,#us,1547,0.06852
#tymoshenko,#ukraine,1533,0.98206

General Look and Feel of Co-Occurrence Visualizations:


OII Network Visualisation Example
file:///Users/AsoSako/Desktop/project2/visualizations/sigmajs/co-occurrence/index.html
Co-occurrence
Co-occurrence
More about this visualisation
Legend:
Co-occurrence
Count
Search:
Search by name
Group Selector:
Select Group
#russia #ukraine
Oxford Internet Institute
University of Oxford
JISC

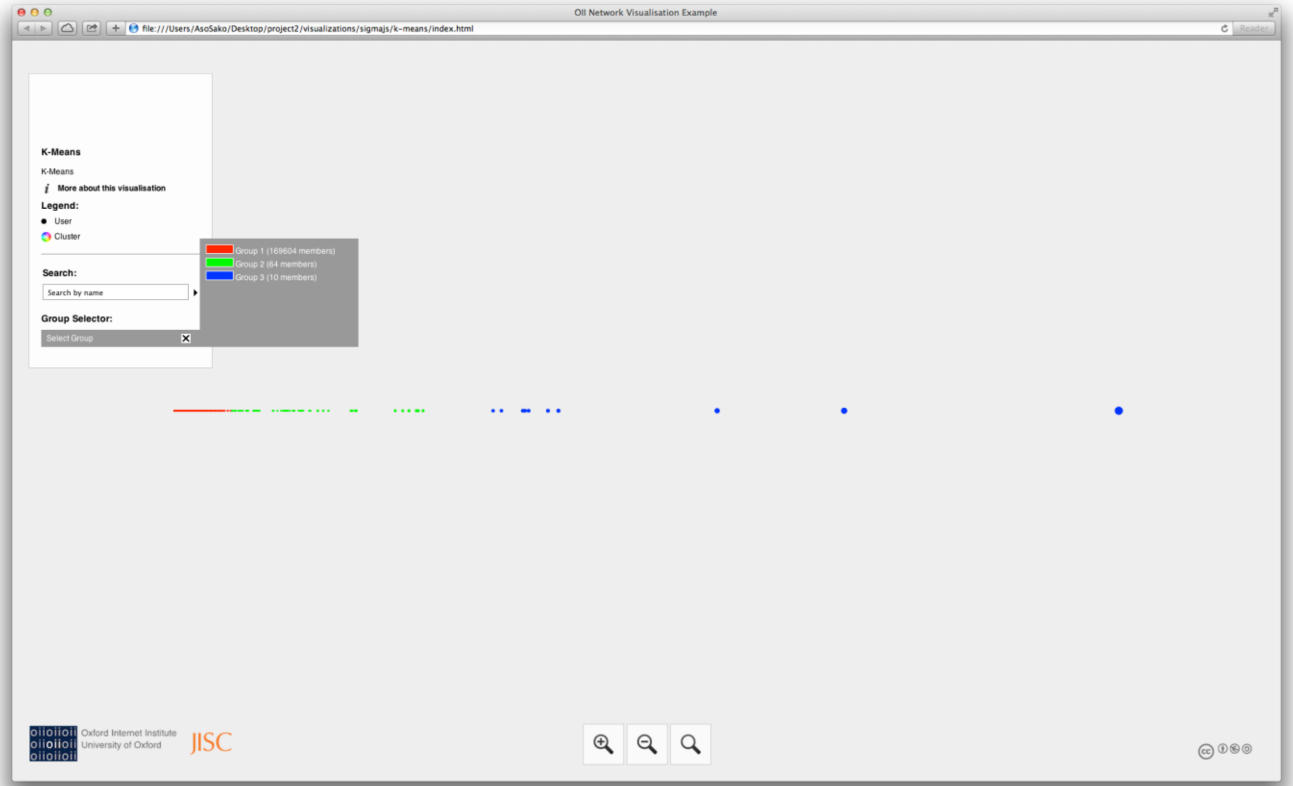
9.3 - K-Means

Top 10 High:
```
cnnbrk,HIGH,16205194
nytimes,HIGH,11493325
bbcbreaking,HIGH,9314252
breakingnews,HIGH,6591304
uberfacts,HIGH,6411873
omgfacts,HIGH,6081104
bbcworld,HIGH,6018512
perezhilton,HIGH,5980331
time,HIGH,5614277
peoplemag,HIGH,5471253
```

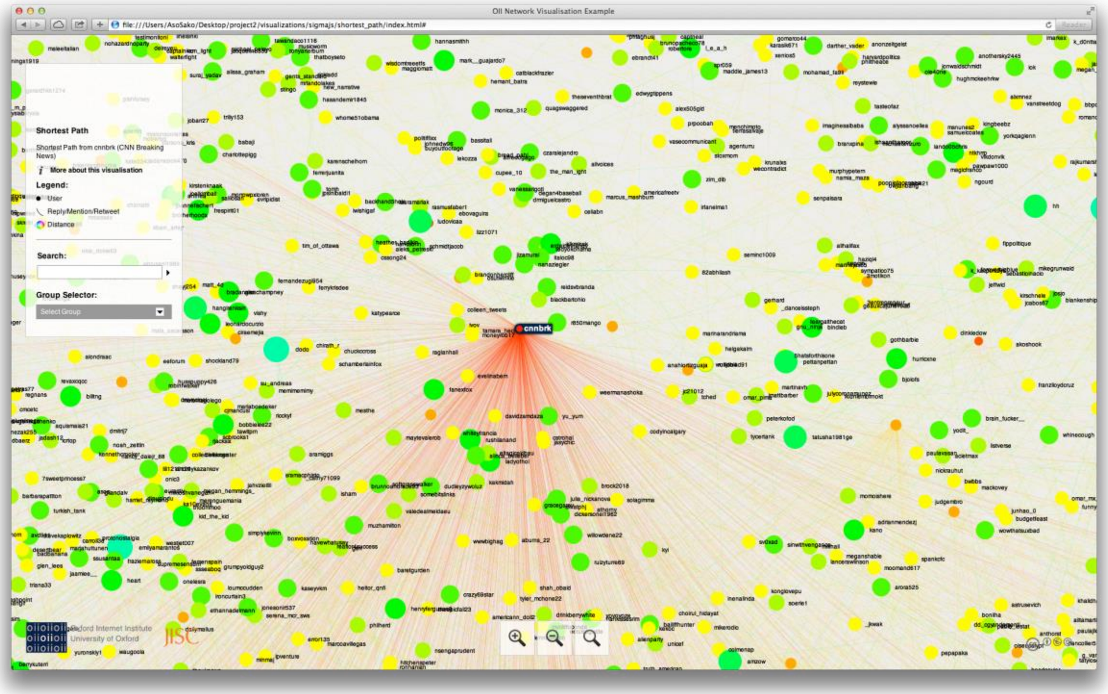
Top 10 Medium:
```
wsj,MEDIUM,4269897
whattheffacts,MEDIUM,4187939
reuters,MEDIUM,4154315
huffingtonpost,MEDIUM,4027565
mashable,MEDIUM,3920122
foxnews,MEDIUM,3792041
ap,MEDIUM,3127941
abc,MEDIUM,3103971
newyorker,MEDIUM,3056099
cbsnews,MEDIUM,3029692
```

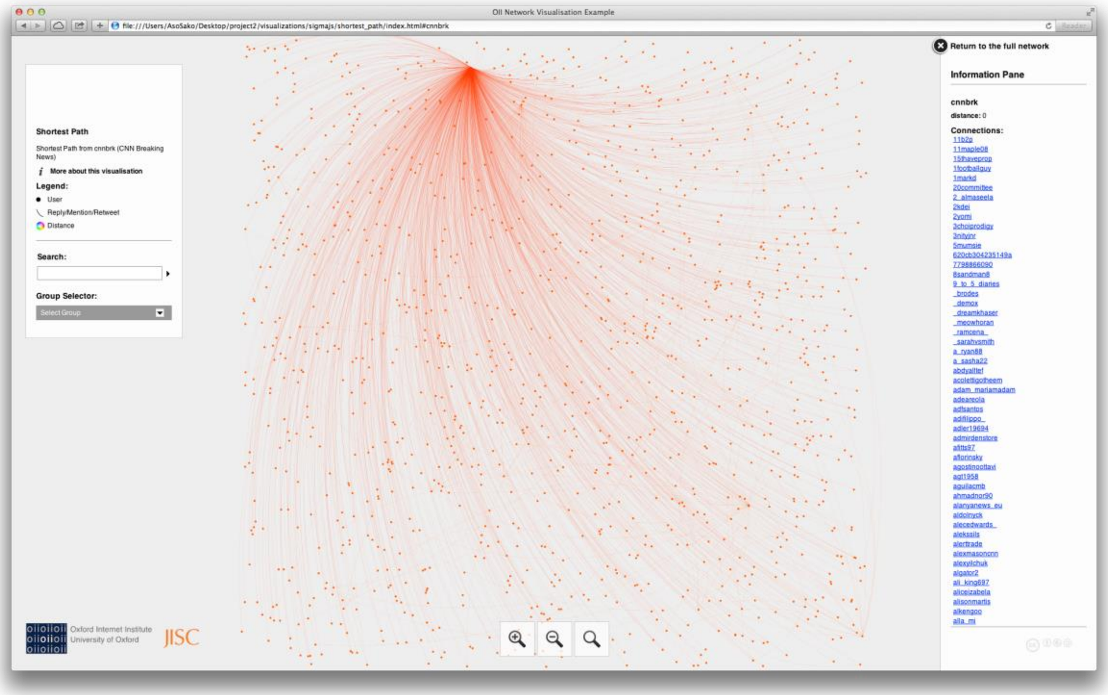
Top 10 Low:
```
treadstone71llc,LOW,967004
mstupenengo,LOW,927385
gopro,LOW,913622
gizmodo,LOW,870848
factboook,LOW,858548
statedept,LOW,856510
buzzfeed,LOW,853798
nbcsnl,LOW,853193
marilynmonroedc,LOW,841597
reutersbiz,LOW,841428
```

General Look and Feel of K-Means Visualizations:



9.4 - Shortest Path (From CNN Breaking News – Top High Group User)


OII Network Visualisation Example
file:///Users/AsoSako/Desktop/project2/visualizations/sigmajs/shortest_path/index.html#
Shortest Path
Shortest Path from cnnbrk (CNN Breaking News)
More about this visualisation
Legend:
User
Reply/Mention/Retweet
Distance
Search:
Group Selector:
Select Group
cnnbrk
Oxford Internet Institute
University of Oxford
JISC


OII Network Visualisation Example
file:///Users/AsoSako/Desktop/project2/visualizations/sigmajs/shortest_path/index.html#cnnbrk
Reader
Oxford Internet Institute University of Oxford
JISC

Shortest Path

Shortest Path from cnnbrk (CNN Breaking News)

i **More about this visualisation**

Legend:

- ● User
- Reply/Mention/Retweet
- Distance

Search:

Group Selector: Select Group

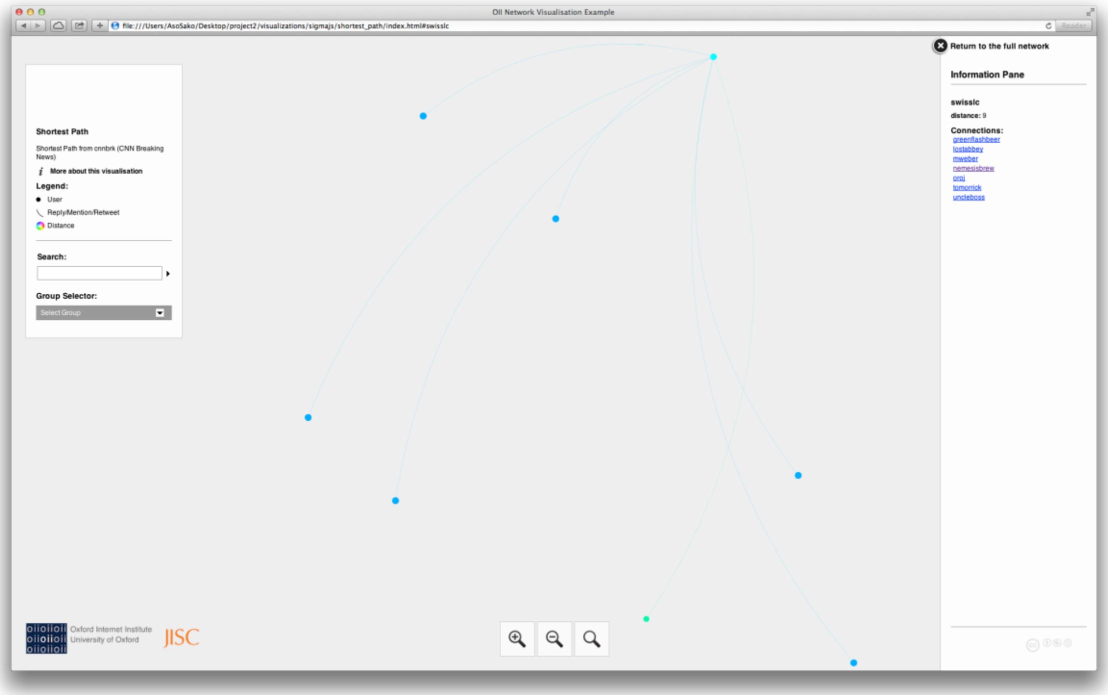
Return to the full network

Information Pane

cnnbrk

distance: 0

Connections:
11b2p
11maple08
15thaveprop
1footballguy
1markd
20committee
2_almaseela
2kdei
2yomi
3choiprodigy
3nityjnr
5mumsie
620cb304235149a
7798866090
8sandman8
9_to_5_diaries
_brodes
_demox
_dreamkhaser
_meowhoran
ramcena
_sarahvsmith
a_ryan88
a_sasha22
abdyalltef
acolettigotheem
adam_mariamadam
adearecia
adfsantos
adifilippo_
adler19694
admirdenstore
afitts97
aflorinsky
agostinoottavi
agt1958
aguilacmb
ahmadnor90
alanyanews_eu
aldoinyck
alecedwards_
aleksails
alerttrade
alexmasoncnn
alexyilchuk
algator2
ali_king697
aliceizabela
alisonmartis
alkengoo
alla_mi


OII Network Visualisation Example
file:///Users/AsoSako/Desktop/project2/visualizations/sigmajs/shortest_path/index.html#swisslc
Reader
Oxford Internet Institute University of Oxford
JISC

Shortest Path

Shortest Path from cnnbrk (CNN Breaking News)

i **More about this visualisation**

Legend:

- ● User
- Reply/Mention/Retweet
- Distance

Search:

Group Selector: Select Group

Return to the full network

Information Pane

swisslc

distance: 9

Connections:
greenflashbeer
lostabbey
mweber
nemesisbrew
orpj
tomorrick
uncleboss

IMPORTANT: For more please check out /visualizations/sigmajs/ or if you want to use Gephi import the graph files from /visualizations/gdf/

Readme for Visualizations:

This readme can also be found in the README.txt in the /visualizations/ directory:

```
------------------------------------------------------------------------------
This folder contains the following files:
------------------------------------------------------------------------------
to_gdf/ - python scripts for converting outputs to Gephi graphs
to_gdf/wordcount.py - transforms wordcount output to gdf
to_gdf/co-occurrence.py - transforms co-occurrence output to gdf
to_gdf/k-means.py - transforms k-means output to gdf
to_gdf/shortest_path.py - transforms shortest path output to gdf
to_gdf/to_gdf.sh - Creates all of the Gephi graph files for this project
gdf/ - Gephi graphs (result of the scripts in to_gdf)
sigmajs/ - Sigmajs Exporter output from Gephi
screenshots/ - select screenshots from the Sigma.js visualizations

------------------------------------------------------------------------------
To fully utilize the visualizations you need to have the following installed:
------------------------------------------------------------------------------
Python 2.7.x
Gephi
Sigmajs Exporter

------------------------------------------------------------------------------
To install the above you need to do the following:
------------------------------------------------------------------------------
1. Download and install Python 2.7.x from: https://www.python.org/downloads
2. Download and install Gephi from: https://gephi.org/
3. Download Sigmajs Exporter from:
   https://marketplace.gephi.org/plugin/sigmajs-exporter/
4. Open Gephi
5. Go to Tools -> Plugins -> Downloaded
6. Click on Add Plugins and add Sigmajs Exporter
7. Click on Install

------------------------------------------------------------------------------
To run the reproduce the Graph file output in gdf/ using the scripts:
------------------------------------------------------------------------------
Make sure the directory structure is not changed for this package.

1. Change your directory to: to_gdf/
2. Run: chmod 777 to_gdf.sh
3. Run: ./to_gdf

------------------------------------------------------------------------------
To import/export graphs from Gephi:
------------------------------------------------------------------------------
1. Open Gephi
2. Click on File -> Open
3. Select the desired graph file
4. Click OK
5. Go to on File -> Export -> Sigma.js template
6. Click on Browse and select a folder to export to
7. Click OK

------------------------------------------------------------------------------
IMPORTANT:
------------------------------------------------------------------------------
It seems Google Chrome has trouble displaying the Sigma.js visualizations
however I have tested it and confirmed it to work in Safari and Firefox.
```

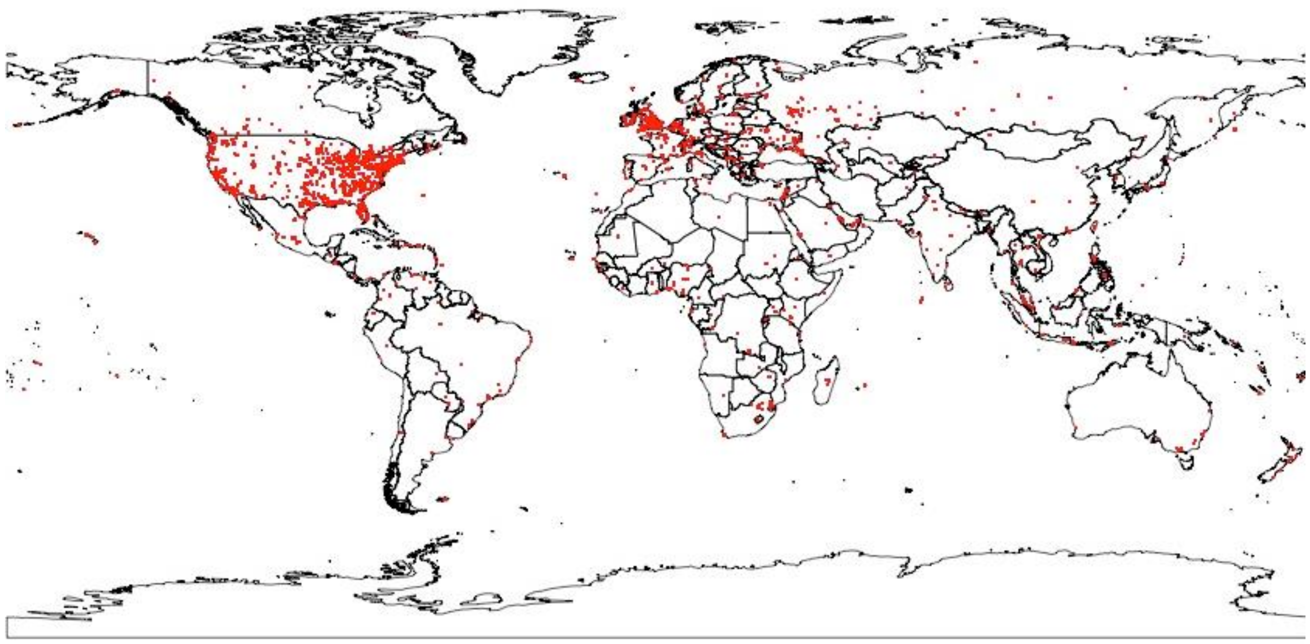
10. Extra (Creative)

Just for fun, for a creative venture, I have extracted coordinates for tweets so that I can plot them using R. I will provide the instructions on them which can be found in the README.txt in the /extra/ folder. I will also provide the plot here, which is in that same directory as well.

```
--------------------------------------------------------------------------------
This folder contains the following:
--------------------------------------------------------------------------------
coordinates.csv - coordinates of tweet origin (few provided)
coordinates.r - creates a plot on the world map for the coordinates
coordinates.jpeg - the above mentioned plot

--------------------------------------------------------------------------------
coordinates.csv format:
--------------------------------------------------------------------------------
The coordinates are collected in 'coordinates.csv' in two columns the first with
longitudes and the second with latitudes.  This data is intended to be used
separately due to being rarely provided along with the tweet.  It is collected
as a creative venture and will not be used during the MR phase of this project.

--------------------------------------------------------------------------------
To run coordinates.r:
--------------------------------------------------------------------------------
1. Install R or R Studio
2. Using R or R Studio run: source('<location_to_file>/coordinates.r')
```



11. Lessons Learned

The Big-Data Content Retrieval, Storage, and Analysis project has taught me many valuable lessons that I am definitely going to take advantage of. First and foremost I learned more about data wrangling, which is an extremely useful skill to have. Using APIs has become a necessity in the sort of big data world we have today. I also got better at Python, which is a good skill to have. More importantly I got one fantastic introduction to Hadoop Map Reduce. I now feel extremely comfortable with the language and methodologies associated with it. In terms of dta analysis I managed to learn some really cool ways to graph large data using Gephi and Sigma.js. Web enabled visualization is truly valuable because it gives one a much better way to browse and understand what is going on.

I would say that this project has been extremely valuable for me. I have found that I can achieve many cool things with ease using Map-Reduce. More importantly I have acquired new ideas for personal projects. It is safe to say that I will definitely dabble with Hadoop again in the future.

The Ukraine Crisis data has been interesting to analyze. The most trending words and hashtags have been fairly predictable. It seems CNN Breaking News is the biggest source of information as it stands. Predictably most users don't have too many followers.