# Geographical Visualization of Tweets

*Harish Gonnabattula, Achyuth Diwakar, Vishal Surya*

CS 242 Information Retrieval and Web Search

Winter 2018

## Introduction

The world is an inter-connected system with edges connecting different cities. Everyday there is Petabytes of data being generated which when properly analyzed provides some interesting patterns. In our project we aim to analyze some of these patterns which span over different geographical locations(geo-spatial data) and provide various insights.

A topic of interest in one part of the globe can have an affect at different part, thereby generating cause-affect relationship. We have gathered thousands of tweets of trending topics over globe over a period of 7 days and plan to discover a pattern among them such that when searched for a topic, the application would display the results as a sentiment map.

## Collaboration

We contributed equally to this project, with most of the work done through pair programming in group meetings.

Harish:

- Designed the architecture for the Web Ui and search system.
- Implemented the HTML page using Angular Js.
- Create a web server in Java using GlassFish.
- Perform sentiment analysis on the code.

Achyuth:

- Choosing MongoDB over traditional text files as it offers easy storage of documents/JSON objects which are essentially our data.
- Understanding Apache Lucene search; how to and what implement for our project.
- Deciding the Analyzer to use for the QueryParser.
- Analyzing the design and architecture required to retrieve index files in Lucene.
- Implementing Lucene Search Code.

Vishal:

- Reading the Json document and create Inverted Index from Map Reduce tasks.
- Implement Hadoop Search architecture.
- Ranking on the documents retrieved by applying Tf*iDF scoring algorithm.

## Architecture:

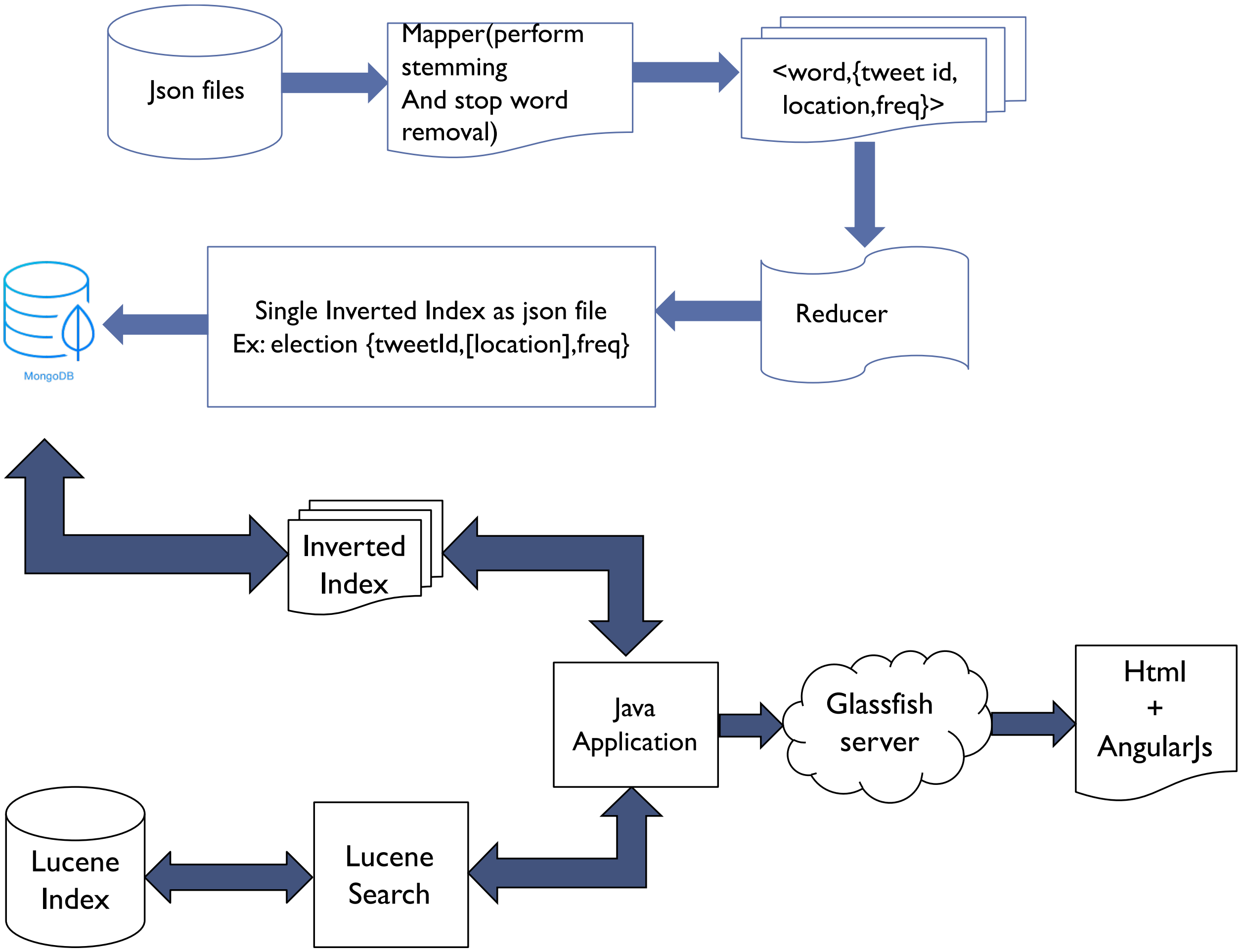
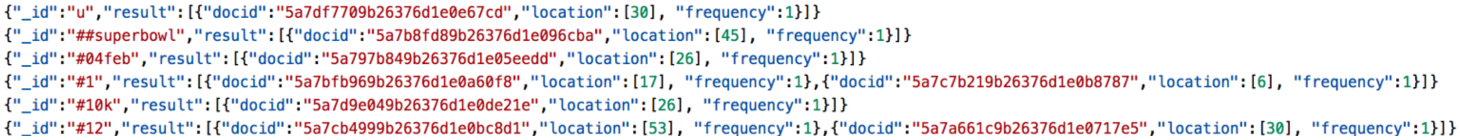

Json files
Mapper(perform stemming And stop word removal)
<word,{tweet id, location,freq}>
Reducer
Single Inverted Index as json file
Ex: election {tweetId,[location],freq}
MongoDB
Inverted Index
Java Application
Glassfish server
Html + AngularJs
Lucene Index
Lucene Search

Figure 1

## Hadoop Inverted Index Construction:

- Data crawled from Twitter is stored in MongoDB as Json objects.
- Export the collection from MongoDB into a Json file as it is easier for the Map Reduce Task to read from a file rather than to read from DB.
- Input to MR task is a Json file of size 5GB. Each object is represented in one line.
- The MR task reads the json object line by line. In our case, each line is one json object.

- From the Tweet Json Object we retrieve the tweet body(this is the text we are indexing) and hashtags.
- The text is first stripped of Stop Words and Stemming is performed on it. This cleans the text.
- Now the text is tokenized and for each token we calculating its frequency in the text body along with its location.
- So we end up having "Token" => {docId,[location],frequency}; **Fig 2.**
- Finally, the output from Map Function is key value pair of format <Text, Text>.
- The reduce function gets a sorted list of words and their respective documents in which they occur along with other meta data.
- Its a simple task in reducer as we have to join the different values of a token and form a Json string.
- Multiple Json strings are clubbed and together outputted as a Json file.



```
{"_id":"u","result":[{"docid":"5a7df7709b26376d1e0e67cd","location":[30], "frequency":1}]}
{"_id":"##superbowl","result":[{"docid":"5a7b8fd89b26376d1e096cba","location":[45], "frequency":1}]}
{"_id":"#04feb","result":[{"docid":"5a797b849b26376d1e05eedd","location":[26], "frequency":1}]}
{"_id":"#1","result":[{"docid":"5a7bfb969b26376d1e0a60f8","location":[17], "frequency":1},{"docid":"5a7c7b219b26376d1e0b8787","location":[6], "frequency":1}]}
{"_id":"#10k","result":[{"docid":"5a7d9e049b26376d1e0de21e","location":[26], "frequency":1}]}
{"_id":"#12","result":[{"docid":"5a7cb4999b26376d1e0bc8d1","location":[53], "frequency":1},{"docid":"5a7a661c9b26376d1e0717e5","location":[30], "frequency":1}]}
```

Fig 2.

## Ranking using Inverted Index & Search:

- Storing the inverted index generated in the previous step into MongoDB aids for a faster retrieval of relevant docs for a query string.
- The end user enters a query string of his choice which is then propagated along the UI pipeline until it reaches our Java application.
- This query string is broken down into individual words and search in the inverted index.
- When matched, the inverted index gives a list of meta data info like documents in which this term is occurring.
- Using the document Id we query the originally stored Mongo Store for the complete data.
- Once the documents are fetched, we are ordering the documents based on their Tf*iDF scores. Fig 3.

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}.$$

Tf = term frequency which is already calculated.

iDF = log(Total Docs/Docs of this word)

- The score is assigned to each document and only the top 30 results are returned to the end user to be displayed on the Web page.
- The tweet body is taken and is passed through Stanford's Sentiment Analyzer to decide the overall sentiment of the tweet.

## Lucene Search:

Apache Lucene is an indexing system which is used to index large text files by tokenizing them and removing all the start/stop words. This is very useful while searching using the keywords.

Lucene works by creating an index and writing it to disk or for smaller indexes store it in RAM. After creating the indexes we create objects called as Documents which store all the required fields and tokenize them. Fields are manually added to these documents like <key,value> pairs and these documents are finally added to the Index using IndexWriter.

- Lucene provides inbuilt functions for searching on the indexed data.
- It automatically calculates the similarity scores of the documents using the BM25 algorithm

$$\sum_{i\in Q} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$$

- In the Part A of our projected, we have indexed fields like Tweet body and hashtags which form around 90% of the search queries.
- In Part B, we have provided an user interface for the user to enter query terms to be searched. The query term are searched on the Lucene Index using the inbuilt function QueryParser.
- Query Parser, a lexer which interprets a string into a Lucene Query using JavaCC.

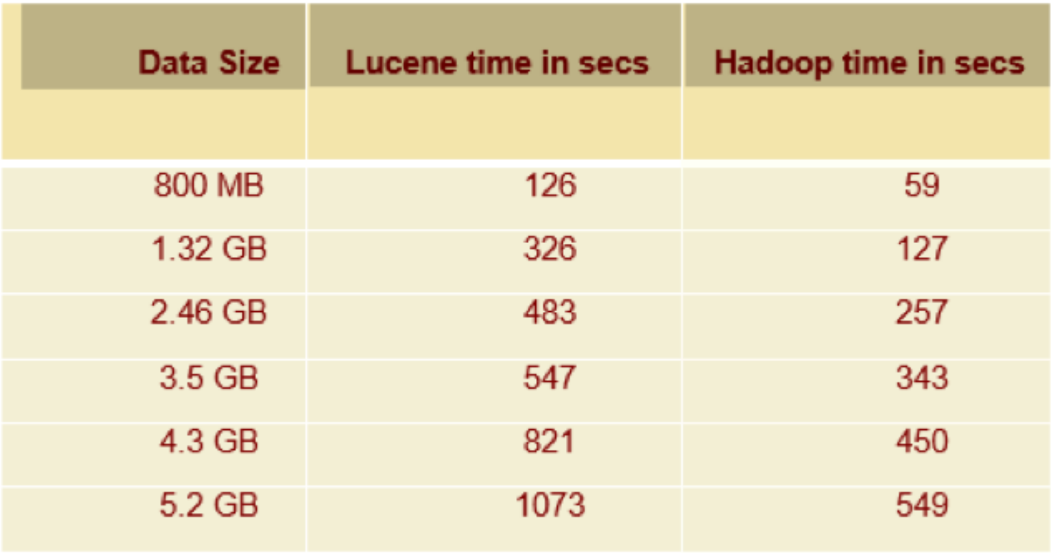
### Lucene Search Code Snippet:

```
Directory indexDirectory = FSDirectory.open(Paths.get(INDEX_PATH));
IndexReader indexReader = DirectoryReader.open(indexDirectory);
final IndexSearcher indexSearcher = new IndexSearcher(indexReader);
//Query Parser for searching the Lucene Index
QueryParser parser = new QueryParser("tweetbody", aWrapper);
```

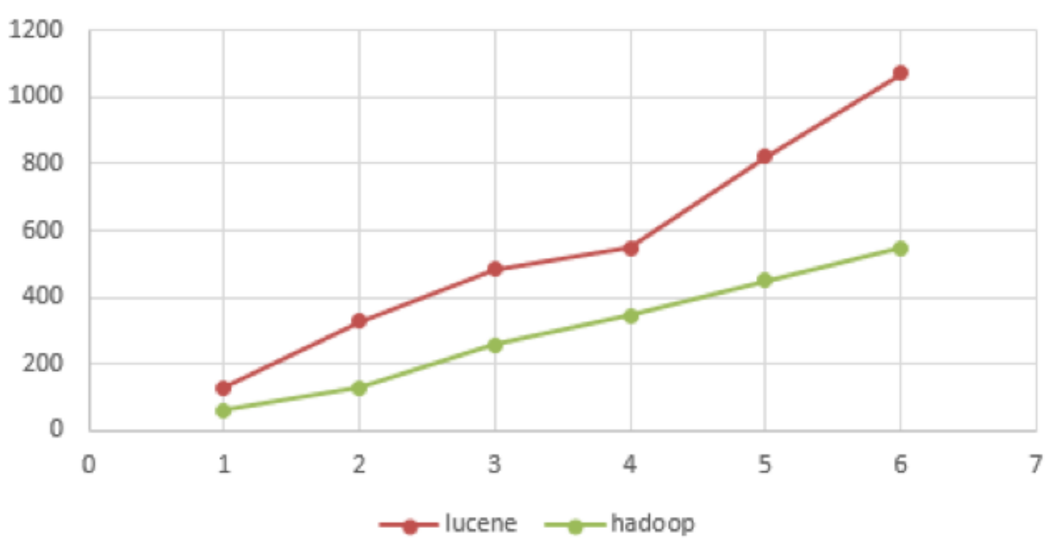
```
Query query = null;

query = parser.parse(keyword);

TopDocs topDocs = indexSearcher.search(query, 30);
ScoreDoc scoreDocs[] = topDocs.scoreDocs;
```

- QueryParser takes two parameters in which the first param is the field in which to search the query for and second is the Analyzer by which the keyword is to be analyzed.
- Next the QueryParser parses the keyword and used indexSearcher's search function to retrieve the documents. We are limiting the results to 30.
- The tweet body is taken and is passed through Stanford's Sentiment Analyzer to decide the overall sentiment of the tweet.

## No.of Documents Indexed vs Time Taken:



| Data Size | Lucene time in secs | Hadoop time in secs |
|---|---|---|
| 800 MB | 126 | 59 |
| 1.32 GB | 326 | 127 |
| 2.46 GB | 483 | 257 |
| 3.5 GB | 547 | 343 |
| 4.3 GB | 821 | 450 |
| 5.2 GB | 1073 | 549 |



## Limitations:

- The sentiment analysis of the tweet does not take into account aspects like, sarcasm and irony and thus it would cause such text to be categorized in the incorrect sentiment.
- The application relies on the geographic location details of the tweet to provide a sentiment trend visualization on the global map. However, it was found that only a small fraction of the tweets provide this information thus affecting the scale of visualization.

## Obstacles:

### Querying more than two words in HADOOP:

Querying for more than one word in Hadoop was a challenge. Querying two words gives a combination of both sets of documents for the keywords that are input for which the scores are calculated. The sum of the tf-idf scores has to be calculated and the documents with the highest scores are returned. This particular case is not handled as it proved to be a challenge.

### Hadoop Indexing:

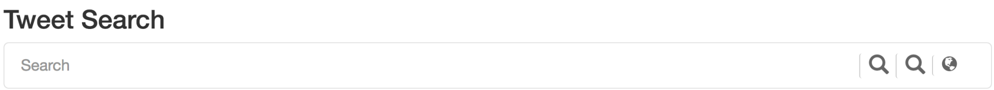
Hadoop MR for indexing gives a sequence file instead of a JSON file which has to be then  converted explicitly into a JSON file and then stored in MongoDB for searching and querying  which was an additional overhead.

## Deployment:

- To run Hadoop Indexer we have provided a .jar file.
- Run the jar file with the Json file as 1st parameter and output directory as 2nd parameter.
- Export the outputted sequence file into a json file and import into MongoDB collection.
- Make sure you have MongoDB instance up and running.

# UI Screenshots:

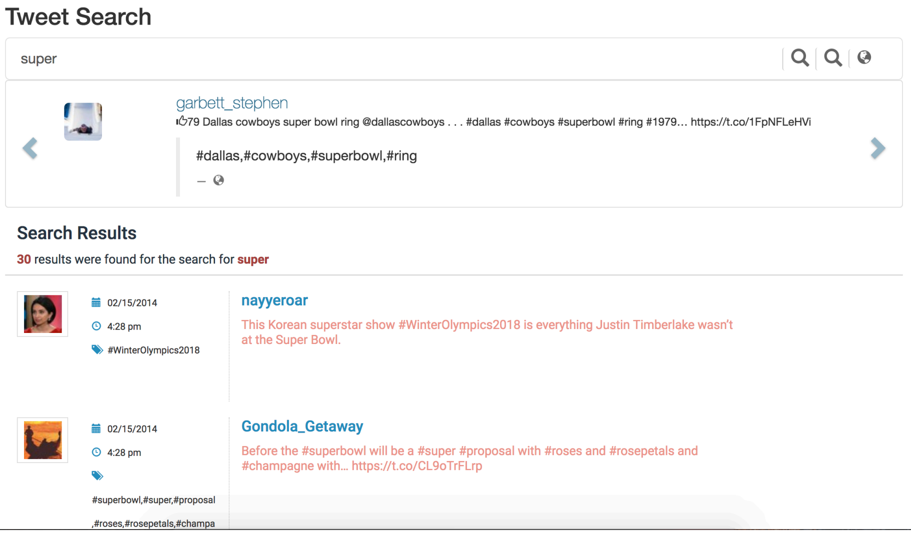
## Homepage:



Tweet Search

Search

## Most Retweeted:



Tweet Search

super

garbett_stephen

79 Dallas cowboys super bowl ring @dallascowboys . . . #dallas #cowboys #superbowl #ring #1979… https://t.co/1FpNFLeHVi

#dallas,#cowboys,#superbowl,#ring

—

Search Results

30 results were found for the search for super

02/15/2014
4:28 pm
#WinterOlympics2018

nayyeroar

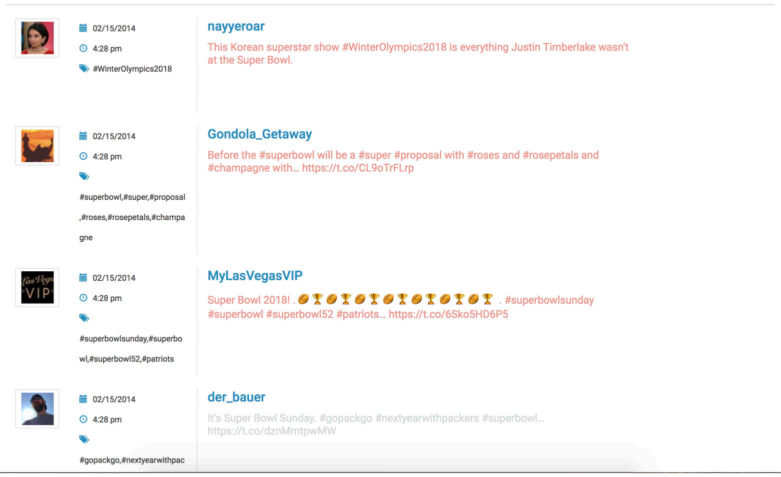
This Korean superstar show #WinterOlympics2018 is everything Justin Timberlake wasn't at the Super Bowl.

02/15/2014
4:28 pm
#superbowl,#super,#proposal
,#roses,#rosepetals,#champa

Gondola_Getaway

Before the #superbowl will be a #super #proposal with #roses and #rosepetals and #champagne with… https://t.co/CL9oTrFLrp

## Search Results with Sentiment Analysis:



02/15/2014
4:28 pm
#WinterOlympics2018

nayyeroar

This Korean superstar show #WinterOlympics2018 is everything Justin Timberlake wasn't at the Super Bowl.

02/15/2014
4:28 pm
#superbowl,#super,#proposal
,#roses,#rosepetals,#champa
gne

Gondola_Getaway

Before the #superbowl will be a #super #proposal with #roses and #rosepetals and #champagne with… https://t.co/CL9oTrFLrp

02/15/2014
4:28 pm
#superbowlsunday,#superbo
wl,#superbowl52,#patriots

MyLasVegasVIP

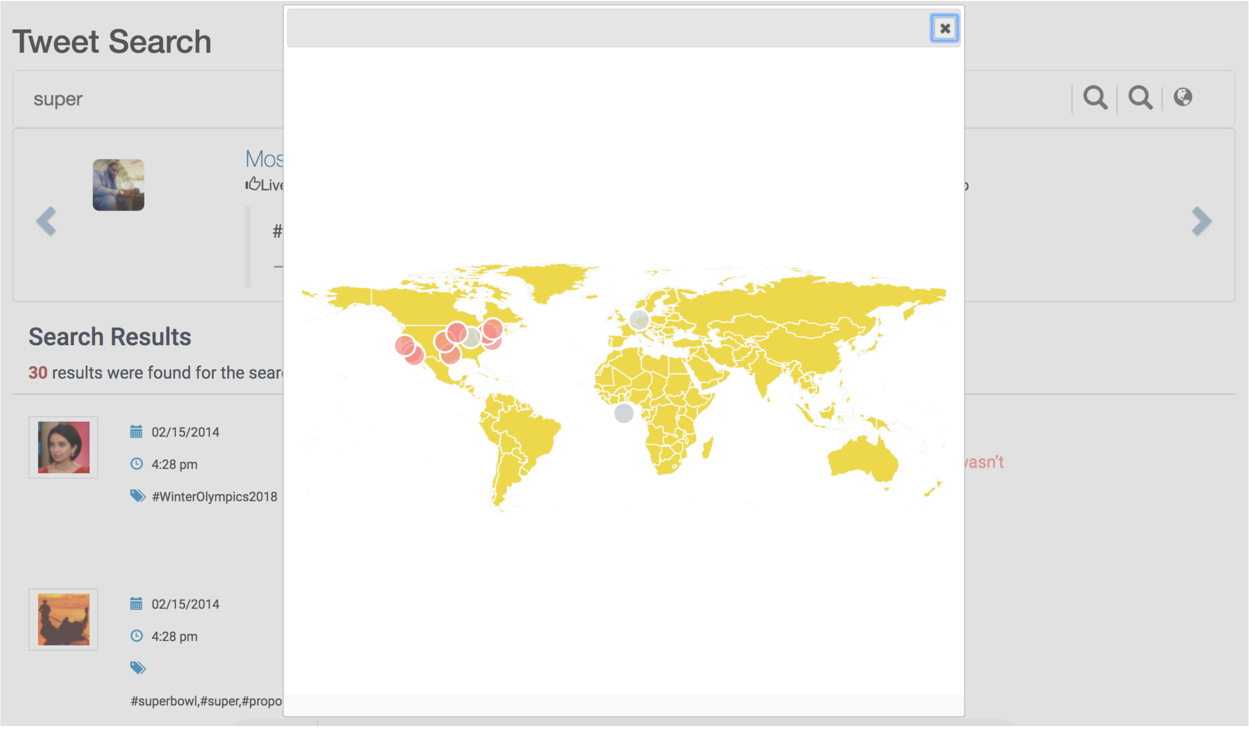
Super Bowl 2018! . 🏉🏆🏉🏆🏉🏆🏉🏆🏉🏆🏉🏆🏉🏆 . #superbowlsunday #superbowl #superbowl52 #patriots… https://t.co/6Sko5HD6P5

02/15/2014
4:28 pm
#gopackgo,#nextyearwithpac

der_bauer

It's Super Bowl Sunday. #gopackgo #nextyearwithpackers #superbowl… https://t.co/dznMmtpwMW

## Visualization:



Tweet Search

super

Mos

Live

#

Search Results

30 results were found for the sear

02/15/2014

4:28 pm

#WinterOlympics2018

vasn't

02/15/2014

4:28 pm

#superbowl,#super,#propo