IDS 561 Analytics for Big data

Project Report

TMiners Twitter Analysis

Mounica Sirineni – 652518967

Nishanth Reddy Konkala – 658824939

Sai Lahari Jalaparthi – 658795799

Contents

1. Introduction:

The goal of this project is to develop a single application to perform two tasks:

I. Visualize the density distribution of tweets across United States map.

II. Sentiment classification of tweets and their distribution.

Our application allows users to enter a keyword and find distribution of tweets about the keyword across United States. The twitter activity in various geographic locations will be shown to the user on an interactive heat map, where users can zoom in and out to interact. Density of tweets can be analyzed through the color intensity of each state on heat map. Users can also obtain the sentiment associated with the keyword. They can view the distribution of public sentiment as positive, negative and neutral.

1.1 Motivation:

In today's time, a microblogging website called Twitter has become an important means of communication. Its services have been found to be faster than conventional news services. People post their opinions on trending topics, current issues or any products as real time messages. All public tweets posted on twitter are freely available through a set of APIs provided by Twitter. This tweet information can be used to find out trending topics online, popularity of a subject in any particular region, and perform statistical analysis.

There are several platforms where user can enter a keyword and search for the tweets. We took the inspiration from www.trendsmap.com, where the global trending tweet tags are displayed on a map. But this webpage will not give the number of tweets or tweet density from different locations. We have designed our application to show the density distribution of tweets on a map. We have also added additional feature to perform sentiment analysis on tweets. Tweets are classified as positive, negative and neutral, which gives the opinion of user. The density distribution provides statistics on how many people are talking about a topic and sentiment analysis provides their opinion on the corresponding topic.

Users on twitter generate over 400 million tweets every day. This large amount of relevant data helps us to achieve a reflection of public statement by analyzing the sentiments expressed in their tweets. Sentiment analysis is important for many applications such as firms to find out the response of their products in the market, predicting political elections and socioeconomic phenomena like stock exchange.

1.2 Technical Challenges:

The Biggest technical difficulty we faced was location identification of the tweet. Initially, the goal of the project was to narrow down the trending topic based on the geo-location of tweets. In order to perform this analysis, we needed geo-referenced tweets. Twitter had only 1% geo referenced tweets and API can access only 1% of the total tweets. Therefore, we were able to get only 0.001% of tweets on which this analysis would not make much sense. Therefore we decided to go with the tweet owner's location instead of tweets exact location assuming that both will be the same.

2. Approach:

The following sequence of steps were followed to build the application:

1) Twitter Authorization
2) Tweets Retrieval
3) Analysis: Once the tweets were retrieved, we performed 2 kinds of analysis on them. They are:
    i. Finding out where a particular key word is trending (Heat maps)
    ii. Sentiment analysis of the tweets containing the key word

2.1. Twitter Authorization:

Initially we need to register the application in Twitter developments apps for the purpose of sending authorized requests to the Twitter API. The registered application will be assigned with the required Consumer Key, Consumer Key Secret and Access Tokens.

Consumer Key: Consumer key is used to identify which application is making the request

Access Token: Access token represents the user's permission to share their account with the application

Process:

1. Login to https://dev.twitter.com/apps with the twitter account
2. Click on 'Create New App' button
3. Fill in the required application details and create the twitter application
4. In the Key and Access Tokens tab, we obtain the Consumer Key and Consumer Key Secret

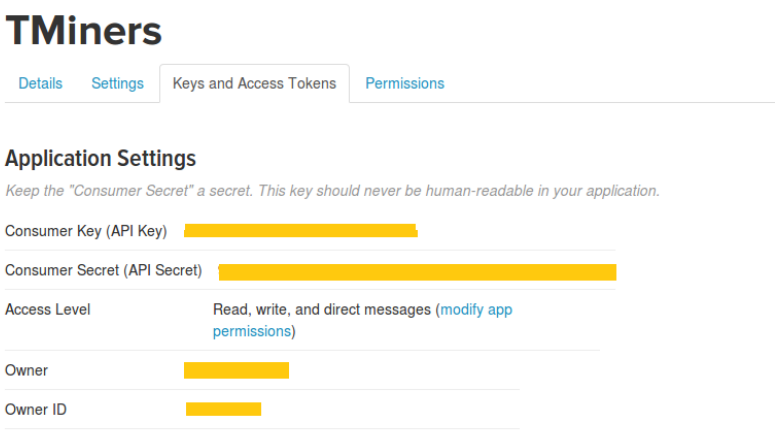


Figure1: Consumer key and secret in application settings

5. Generate the Access Token and Token Secret.

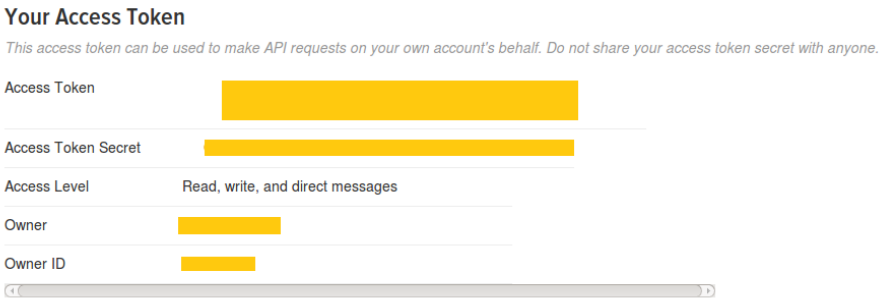


Figure2: Access Token and secret in application settings

2.2. Tweets Retrieval:

We used Twitter4j API to retrieve the tweets using the acquired authorization for the twitter application. Tweets are retrieved for the search keyword entered by user. Below figure is the tweet received in our application.

```
StatusJSONImpl{createdAt=Sat Dec 05 01:03:39 CST 2015, id=673034987029979136,
text='@NikeLasVegas y'all still have any pairs of the cyber Monday air Jordan 1's
10.5-11??', source='<a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>', isTruncated=false, inReplyToStatusId=-1,
inReplyToUserId=430870065, isFavorited=false, isRetweeted=false, favoriteCount=0,
inReplyToScreenName='NikeLasVegas', geoLocation=null, place=null, retweetCount=0,
isPossiblySensitive=false, lang='en', contributorsIDs=[], retweetedStatus=null,
userMentionEntities=[UserMentionEntityJSONImpl{name='Nike Las Vegas',
screenName='NikeLasVegas', id=430870065}], urlEntities=[], hashtagEntities=[],
mediaEntities=[], symbolEntities=[], currentUserRetweetId=-1,
user=UserJSONImpl{id=40596425, name='Glob_Zombie', screenName='Gilbert_Table',
location='Las Vegas', description='§Dabby Waxfingers§Doobie Kushner§Chrispy
Fontaine§Johnny Hash$§Dirty Dabber§Oil Boiler§', isContributorsEnabled=false,
profileImageUrl='http://pbs.twimg.com/profile_images/668910636374470656/hJ-
jY2jO_normal.jpg',
profileImageUrlHttps='https://pbs.twimg.com/profile_images/668910636374470656/hJ-
jY2jO_normal.jpg', isDefaultProfileImage=false, url='null', isProtected=false,
followersCount=191, status=null, profileBackgroundColor='000000',
profileTextColor='5308F5', profileLinkColor='FF0000',
profileSidebarFillColor='C0DFEC', profileSidebarBorderColor='000000',
profileUseBackgroundImage=false, isDefaultProfile=false, showAllInlineMedia=false,
friendsCount=412, createdAt=Sat May 16 22:04:42 CDT 2009, favouritesCount=782,
utcOffset=-32400, timeZone='Alaska',
profileBackgroundImageUrl='http://pbs.twimg.com/profile_background_images/45662945
5230734336/Aq2NLWrJ.jpeg',
profileBackgroundImageUrlHttps='https://pbs.twimg.com/profile_background_images/45
6629455230734336/Aq2NLWrJ.jpeg', profileBackgroundTiled=false, lang='en',
statusesCount=2470, isGeoEnabled=true, isVerified=false, translator=false,
listedCount=1, isFollowRequestSent=false}}
```


Figure3: Tweet retrieved in our application

2.3. Analysis:

2.3.1. Heat Maps:

In order to identify the location of the tweets, we developed our own approach by analyzing the tweets.

Pre-Analysis:

In our pre-analysis by looking into the different location values of the tweets retrieved, we observed that the user location is represented mostly in the below formats

- State, State Abbreviation
- State Abbreviation, State
- City, State Abbreviation
- City

For very few tweets, we can identify the city location. So we decided to analyze the tweet distribution based on state.

State and Cities in US stored in Database:

We used MongoDB to store the state and city details in United States. In one collection, we stored state and state abbreviation codes as document for each state in U.S. In another collection, we stored city and its state as documents for each city in U.S.

```
> db.stateCity.find()
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0cf"), "City" : "Anderson", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d0"), "City" : "Anchorage", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d1"), "City" : "Angoon", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d2"), "City" : "Atqasuk", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d3"), "City" : "Barrow", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d4"), "City" : "Bethel", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d5"), "City" : "Clear", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d6"), "City" : "Cordova", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d7"), "City" : "Craig", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d8"), "City" : "Delta Junction", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0d9"), "City" : "Dillingham", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0da"), "City" : "Eielson Air Force Base", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0db"), "City" : "Elmendorf Air Force Base", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0dc"), "City" : "Fairbanks", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0dd"), "City" : "Galena", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0de"), "City" : "Glennallen", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0df"), "City" : "Haines", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0e0"), "City" : "Healy", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0e1"), "City" : "Hoonah", "State" : "Alaska" }
{ "_id" : ObjectId("5642d87c3796ee33c6a1e0e2"), "City" : "Juneau", "State" : "Alaska" }
```

Figure4: City and State value stored in MongoDB

```
> db.state_abr.find()
{ "_id" : ObjectId("5642d7d33796ee33c6a1e09d"), "State" : "Alabama", "Abbrevation" : "AL" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e09e"), "State" : "Alaska", "Abbrevation" : "AK" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e09f"), "State" : "Arizona", "Abbrevation" : "AZ" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a0"), "State" : "Arkansas", "Abbrevation" : "AR" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a1"), "State" : "California", "Abbrevation" : "CA" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a2"), "State" : "Colorado", "Abbrevation" : "CO" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a3"), "State" : "Connecticut", "Abbrevation" : "CT" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a4"), "State" : "Delaware", "Abbrevation" : "DE" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a5"), "State" : "Florida", "Abbrevation" : "FL" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a6"), "State" : "Georgia", "Abbrevation" : "GA" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a7"), "State" : "Hawaii", "Abbrevation" : "HI" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a8"), "State" : "Idaho", "Abbrevation" : "ID" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0a9"), "State" : "Illinois", "Abbrevation" : "IL" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0aa"), "State" : "Indiana", "Abbrevation" : "IN" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0ab"), "State" : "Iowa", "Abbrevation" : "IA" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0ac"), "State" : "Kansas", "Abbrevation" : "KS" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0ad"), "State" : "Kentucky", "Abbrevation" : "KY" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0ae"), "State" : "Louisiana", "Abbrevation" : "LA" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0af"), "State" : "Maine", "Abbrevation" : "ME" }
{ "_id" : ObjectId("5642d7d33796ee33c6a1e0b0"), "State" : "Maryland", "Abbrevation" : "MD" }
```

Figure5: State and State abbreviation stored in MongoDB

Process:

1. Keyword for the search is taken as input from the user

2. Read the state and cities stored in MongoDB for the validation of the user profile location for each tweet
3. User profile location is obtained for each tweet which is in different formats
4. Identified whether the location is one among the U.S. states by splitting the retrieved user profile location and validate it against the state and city value stored in the database
5. Once the location information is standardized, the location is inserted into MongoDB. The location value is stored in "US – State Abbreviation" format which will be helpful for heat maps

```
> db.tweets.find()
{ "_id" : ObjectId("566292063fe50b2c511a6781"), "location" : "US-AR" }
{ "_id" : ObjectId("566292063fe50b2c511a6783"), "location" : "US-VT" }
{ "_id" : ObjectId("566292063fe50b2c511a6785"), "location" : "US-PA" }
{ "_id" : ObjectId("566292063fe50b2c511a6787"), "location" : "US-PA" }
{ "_id" : ObjectId("566292063fe50b2c511a6789"), "location" : "US-OH" }
{ "_id" : ObjectId("566292063fe50b2c511a678b"), "location" : "US-OH" }
{ "_id" : ObjectId("566292073fe50b2c511a678d"), "location" : "US-OH" }
{ "_id" : ObjectId("566292073fe50b2c511a678f"), "location" : "US-CA" }
{ "_id" : ObjectId("566292073fe50b2c511a6791"), "location" : "US-OH" }
{ "_id" : ObjectId("566292073fe50b2c511a6793"), "location" : "US-NM" }
{ "_id" : ObjectId("566292073fe50b2c511a6795"), "location" : "US-NY" }
{ "_id" : ObjectId("566292083fe50b2c511a6797"), "location" : "US-AR" }
{ "_id" : ObjectId("566292083fe50b2c511a6799"), "location" : "US-CA" }
{ "_id" : ObjectId("566292083fe50b2c511a679b"), "location" : "US-CA" }
{ "_id" : ObjectId("566292083fe50b2c511a679d"), "location" : "US-PA" }
{ "_id" : ObjectId("566292083fe50b2c511a679f"), "location" : "US-CA" }
{ "_id" : ObjectId("566292083fe50b2c511a67a1"), "location" : "US-OH" }
{ "_id" : ObjectId("566292083fe50b2c511a67a3"), "location" : "US-MD" }
{ "_id" : ObjectId("566292083fe50b2c511a67a5"), "location" : "US-VA" }
{ "_id" : ObjectId("566292083fe50b2c511a67a7"), "location" : "US-OH" }
```

Figure6: Location inserted into MongoDB after location identification

6. On the Location information, implemented MapReduce algorithm in Hadoop with input from MongoDB as the MongoInputFormat to the Hadoop environment
7. Output of this process is moved into the project workspace (local system)

```
US-AK   1
US-AL   1
US-AR   2
US-CA   14
US-CO   1
US-FL   3
US-GA   2
US-IA   1
US-IL   1
US-IN   1
US-KY   1
US-MA   1
US-MD   3
US-ME   1
US-MN   11
US-ND   1
US-NM   1
US-NV   3
US-NY   4
US-OH   11
US-OR   1
US-PA   5
US-SD   1
US-TN   1
US-TX   7
```

Figure7: MapReduce output

8. In order to display these results on a heat map, we converted the above output in text format into required JSON format with ID as state and value as count

```
{ "map": "usaLow", "areas": [{"id":"US-AK","value":1},
{"id":"US-AL","value":1},
{"id":"US-AR","value":2},
{"id":"US-CA","value":14},
{"id":"US-CO","value":1},
{"id":"US-FL","value":3},
{"id":"US-GA","value":2},
{"id":"US-IA","value":1},
{"id":"US-IL","value":1},
{"id":"US-IN","value":1},
{"id":"US-KY","value":1},
{"id":"US-MA","value":1},
{"id":"US-MD","value":3},
{"id":"US-ME","value":1},
{"id":"US-MN","value":11},
{"id":"US-ND","value":1},
{"id":"US-NM","value":1},
{"id":"US-NV","value":3},
{"id":"US-NY","value":4},
{"id":"US-OH","value":11},
{"id":"US-OR","value":1},
{"id":"US-PA","value":5},
{"id":"US-SD","value":1},
{"id":"US-TN","value":1},
```

Figure8: MapReduce converted to JSON

9. We used the heat map to show the density distribution to the users. Input to the heat map is the converted JSON file.

Flow Diagram:

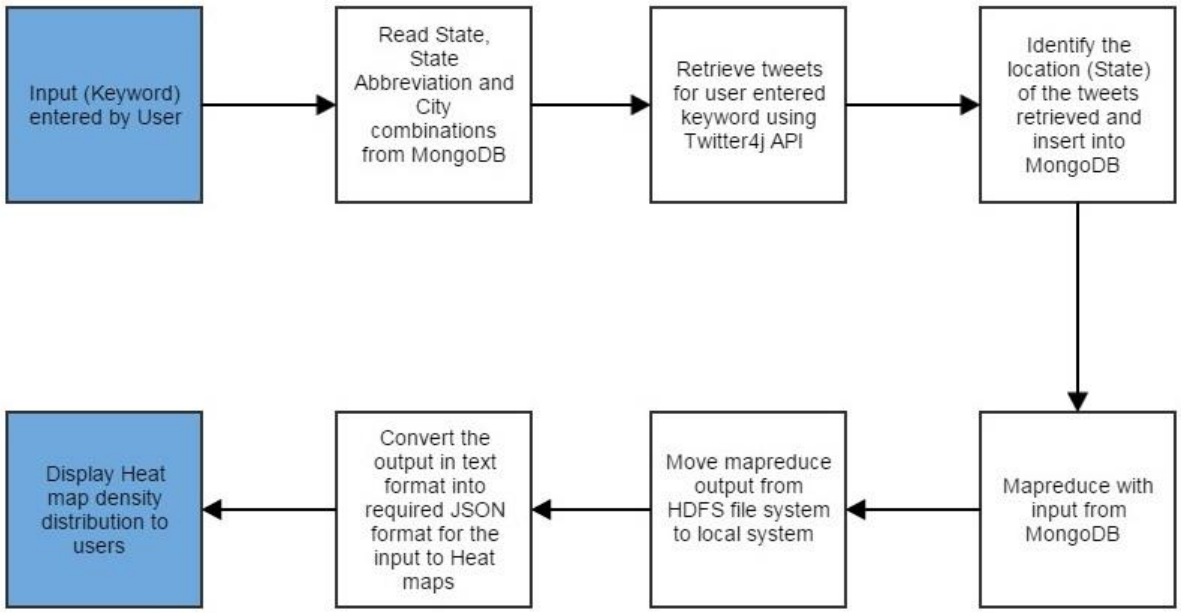


Figure 9: Density Distribution process

In Figure 9, the boxes highlighted in blue color represents the front-end to the user and the remaining are steps implemented back-end.

AmCharts:

In AmCharts, amMap is an open source interactive JavaScript/HTML5 library. AmMaps loads the data either in the JSON or CSV format. In our application we used JSON format as the input to heat maps. AmMaps validates the data which is in format of ID and Value. In our dataset, which is the output of MapReduce, state is the ID and count is the Value. After reading the data, the map will automatically choose color based on its value compared with all the values.

The state with the highest number of tweets is represented in Red color; the state with lowest number of tweets is represented in light yellow color; the states with intermediate values are represented in intermediate colors based on its value; the state which doesn't have any tweets is represented in white color.

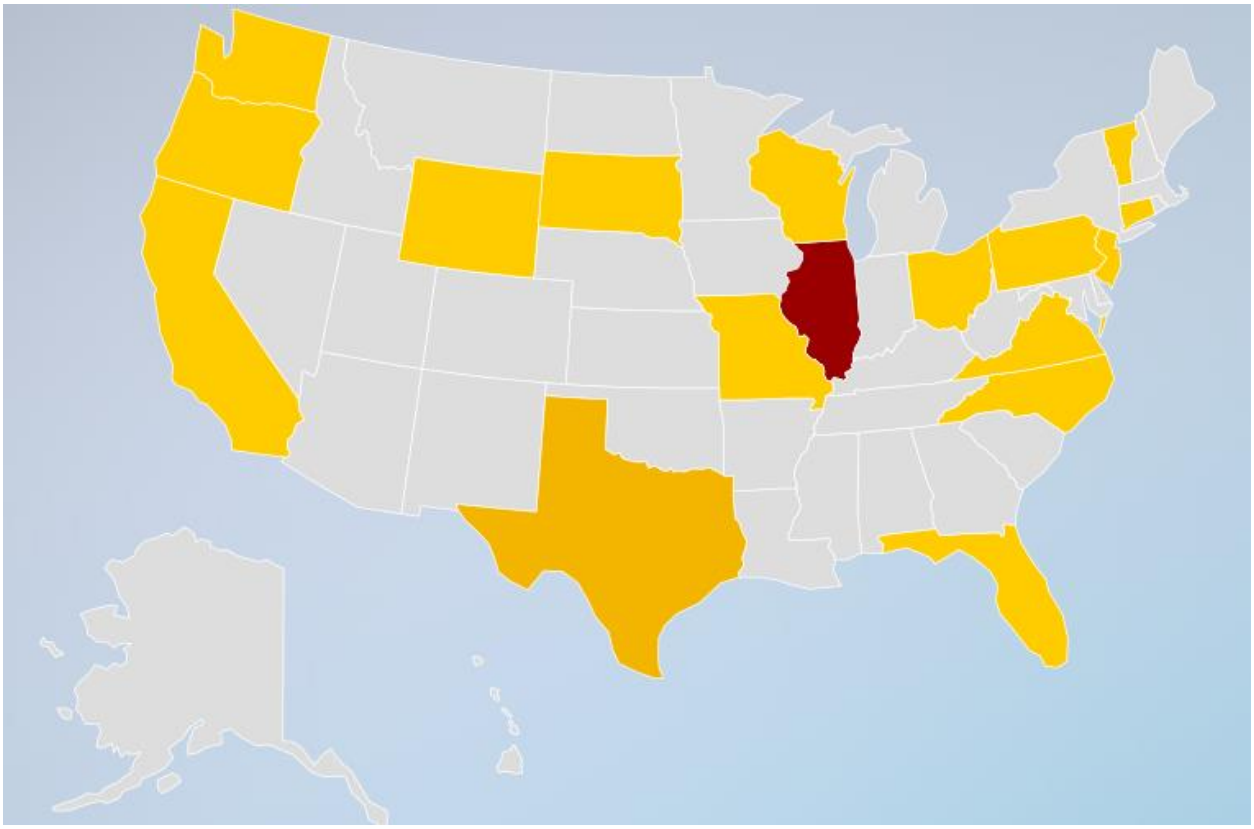


Figure10: Heat maps

2.3.2. Sentiment Analysis:

We tested both Stanford NLP and LingPipe APIs for performing sentiment analysis. While using Stanford NLP, we found that majority (80%) of tweets were given sentiment rating of "1", which is not very informative. Hence we decided to use LingPipe API.

LingPipe is tool kit for processing text using computational linguistics. LingPipe's architecture is designed to be efficient, scalable, reusable, and robust. Using this API, we can train our model and generate a classifier, which suits our needs. A classifier is a file, which contains rules to assign Sentiment rating for a tweet. For this project, we decided to use the default classifier provided with the APIs, which will be able to predict sentiment with 75% accuracy. The API will assign each tweet a sentiment rating of: Positive, Negative or Neutral.

Process:

1) Keyword for the tweets search is taken as input from the user

2) Tweets retrieval, Data pre-processing and insert into MongoDB

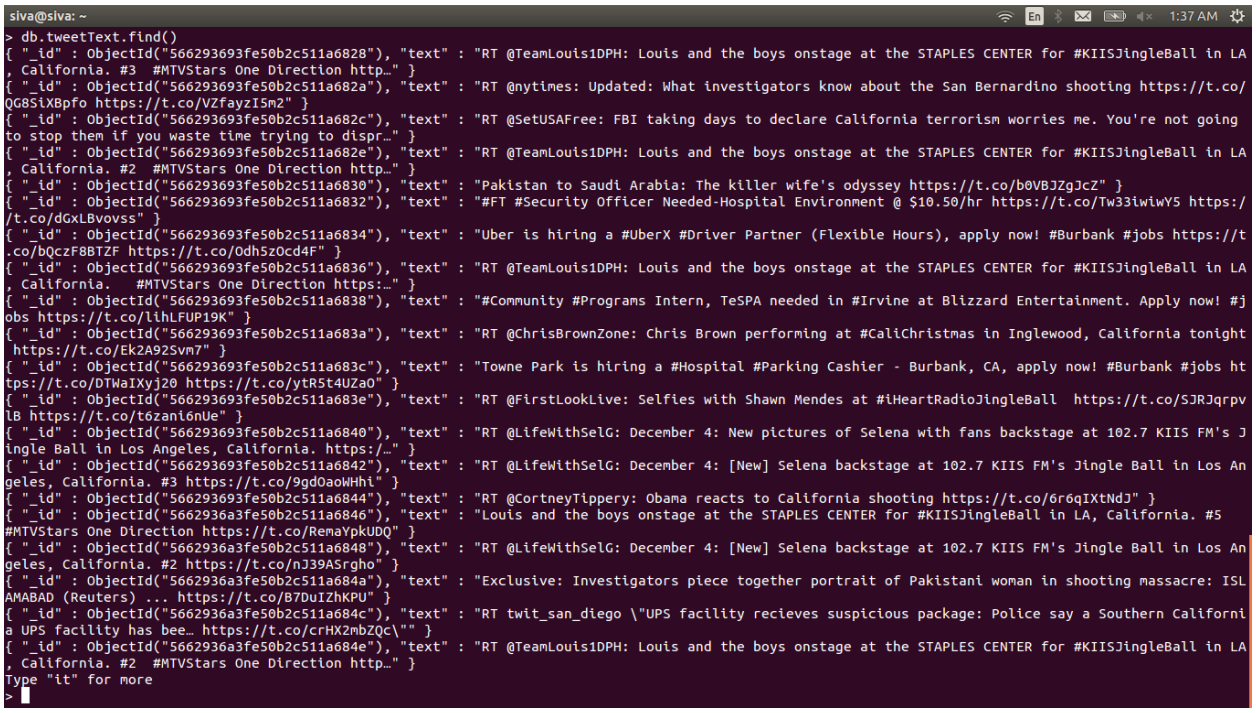


Figure11: Tweet text inserted into MongoDB

3) Implement map reduce algorithm with the tweets as input from MongoDB
4) In the mapper class, we used classification algorithm to classify the tweets into Positive, Negative or Neutral. The classified output is given as input to the reducer class
5) The output from the MapReduce and classification algorithm is moved to the local file system

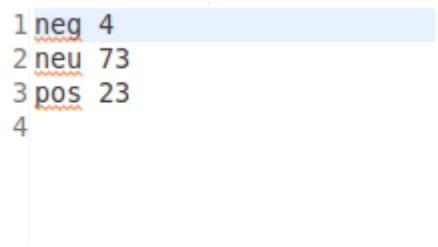


Figure12: MapReduce output after sentiment analysis classification

6) The output from the local file system is read by the java application and converted into percentages
7) Generated pie chart using JFreeChart library for graphical representation of sentiment analysis

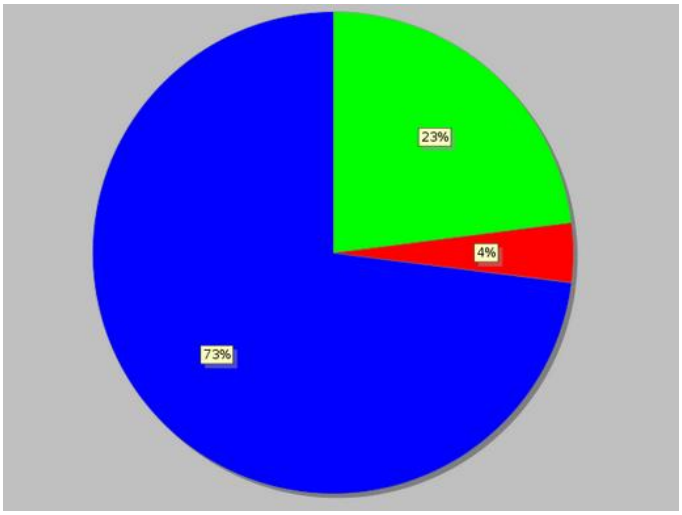


Figure13: MapReduce output converted to percentages and represented in pie chart

Flow Diagram:

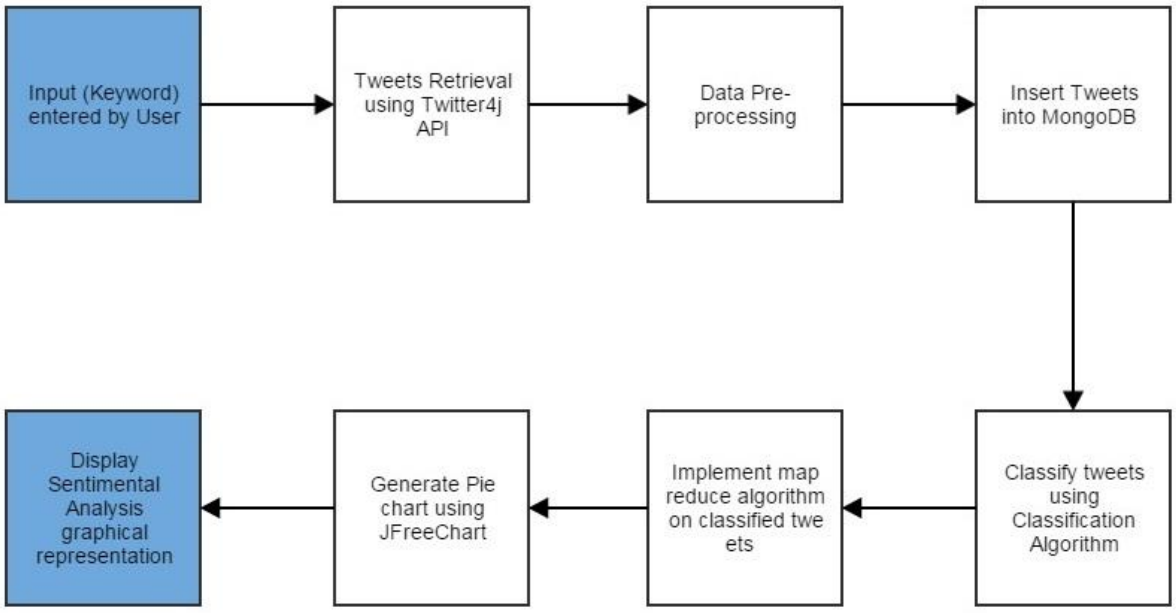


Figure 14: Sentiment analysis process

In Figure 14, the boxes highlighted in blue color represents the front-end to the user and the remaining are steps implemented back-end.

3. Data Description

We used Twitter4j APIs to collect tweets from the twitter data. There are two possible ways to collect the data from Twitter API: “Twitter Search” and “Twitter Stream”. Twitter search will search through existing tweets using Query class, whereas Twitter Stream will consume live tweets that match the keyword. Both APIs have restriction on the number of tweets that can be accessed. Using these APIs, we can access only 1% of total tweets.

API Rate Limits:

The API rate limits can be based on per-user basis or per-application basis. For application-only authentication, rate limit is determined for the entire application. Rate limits are divided into 15 minute windows.

In Twitter Search API, Search will be rate limited to 180 queries per 15min window. If the rate limit is exceeded, the Twitter API will return a code of HTTP 429 representing too many requests.

In Twitter Streaming API, Clients who reconnect frequently with the change in the parameter search every time run the risk of being rate limited. Rate limited clients will receive HTTP 420 responses for all connection requests.

Fetching Tweets:

Our application motive is to use real time data instead of fetching huge number of tweets on selected topics and store them in database. This will also help to enter any keyword by the user instead of the selected topics only. Fetching real time data will take considerable time using Twitter Streaming API than Twitter Search API. We decided to use Twitter Search API, as it takes less time to fetch the tweets. The application can be scaled up to be more efficient by fetching all the tweets without restrictions by using Twitter Firehose API, which allows unlimited access to the twitter information. Partners of Twitter only will have access to Twitter Firehouse API.

Description of Tweets:

Tweets known as ‘Status Updates’, are short length messages with a maximum length of 140 characters. Users use a lot of acronyms, hashtags, emoticons, slang and special characters. @ Operator is used to refer to other users. # called as hashtag is used to mark keywords or topics in the tweet.

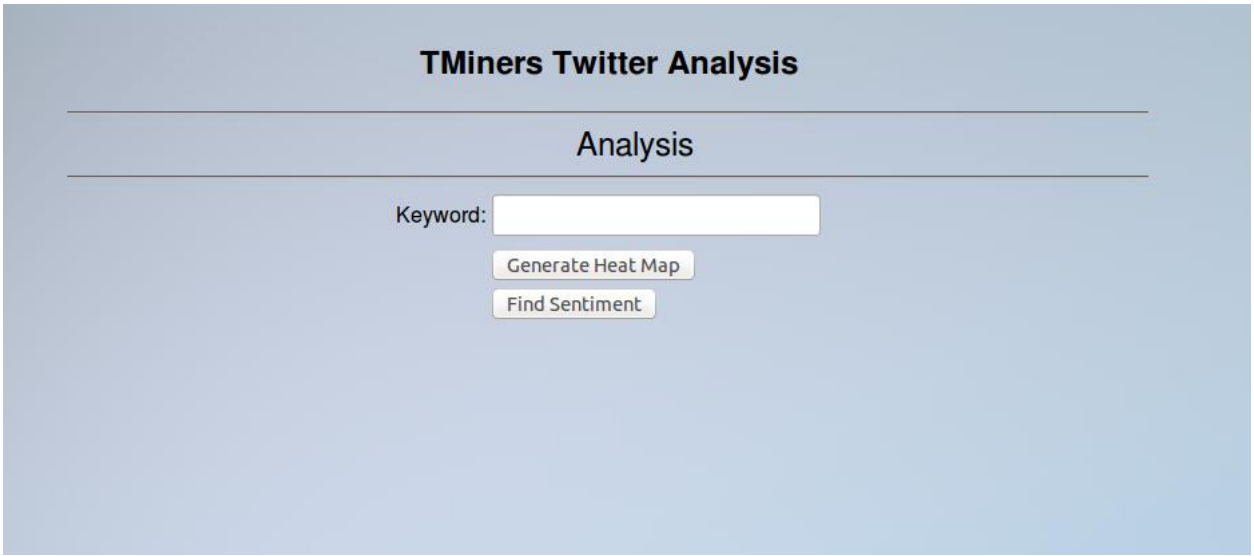
Users also post URLs of WebPages to share information. Acronyms are mostly used by users due to limitation of characters.

Below are the two fields that are used for our analysis

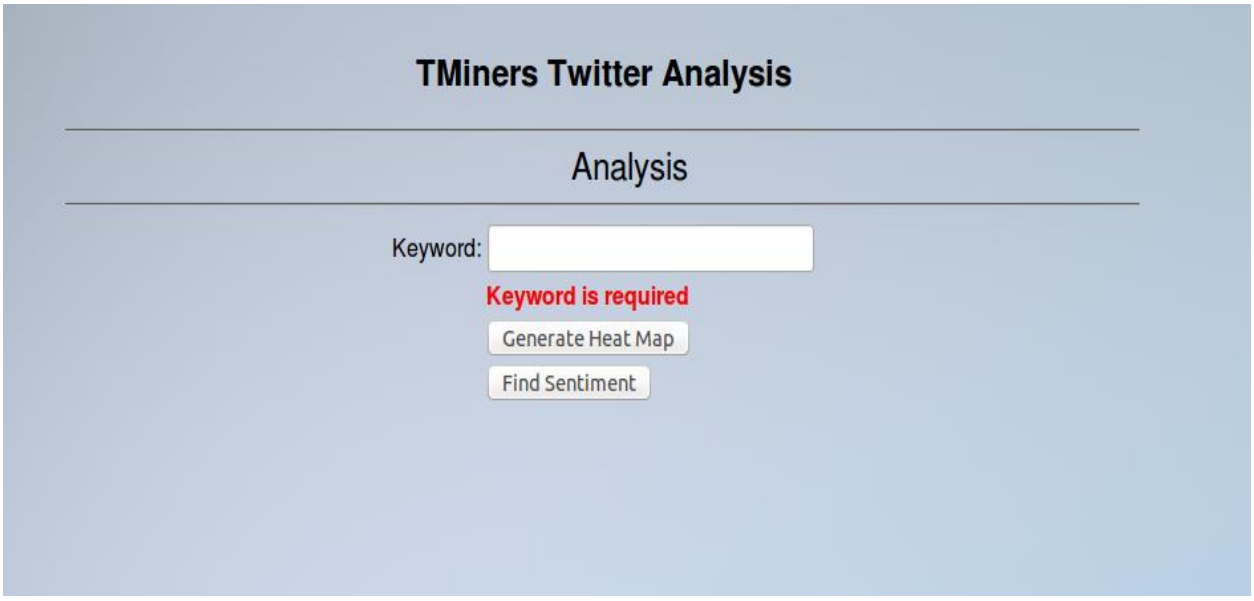
Field	Type	Description
User	Users	The user who posted the tweet. From this field, we can get the user location.
text	String	The actual UTF-8 text of the status update.

Table 1: Fields used in the analysis

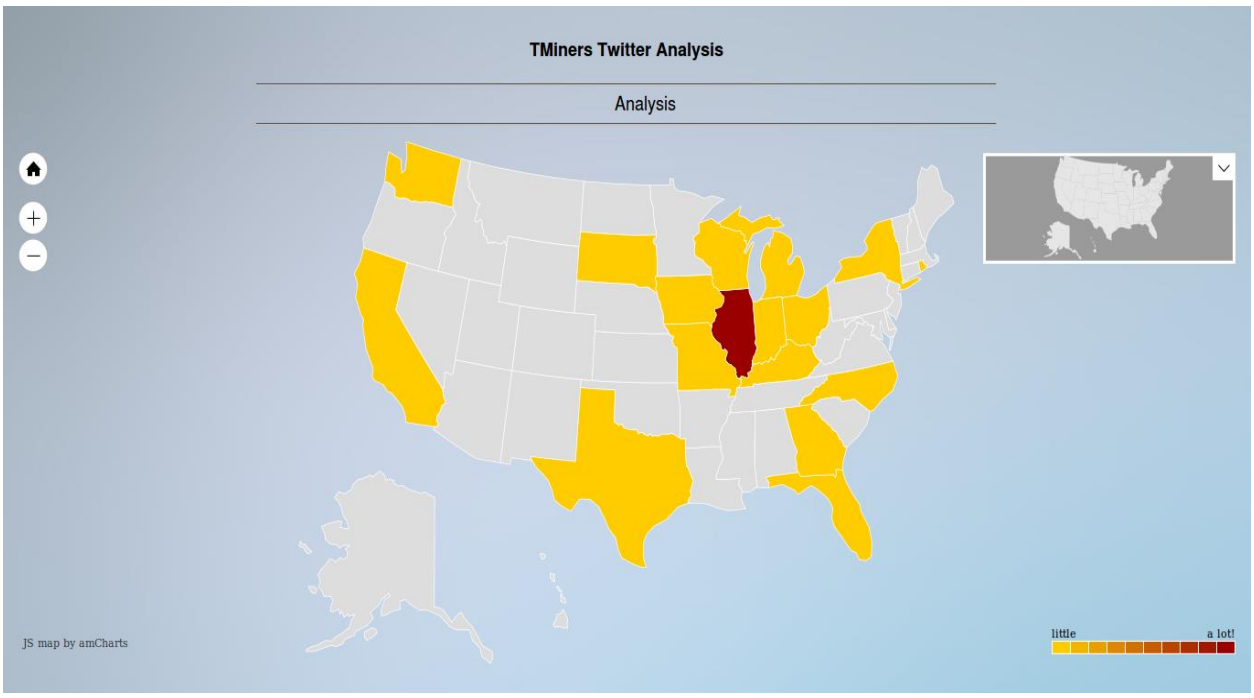
4. Results



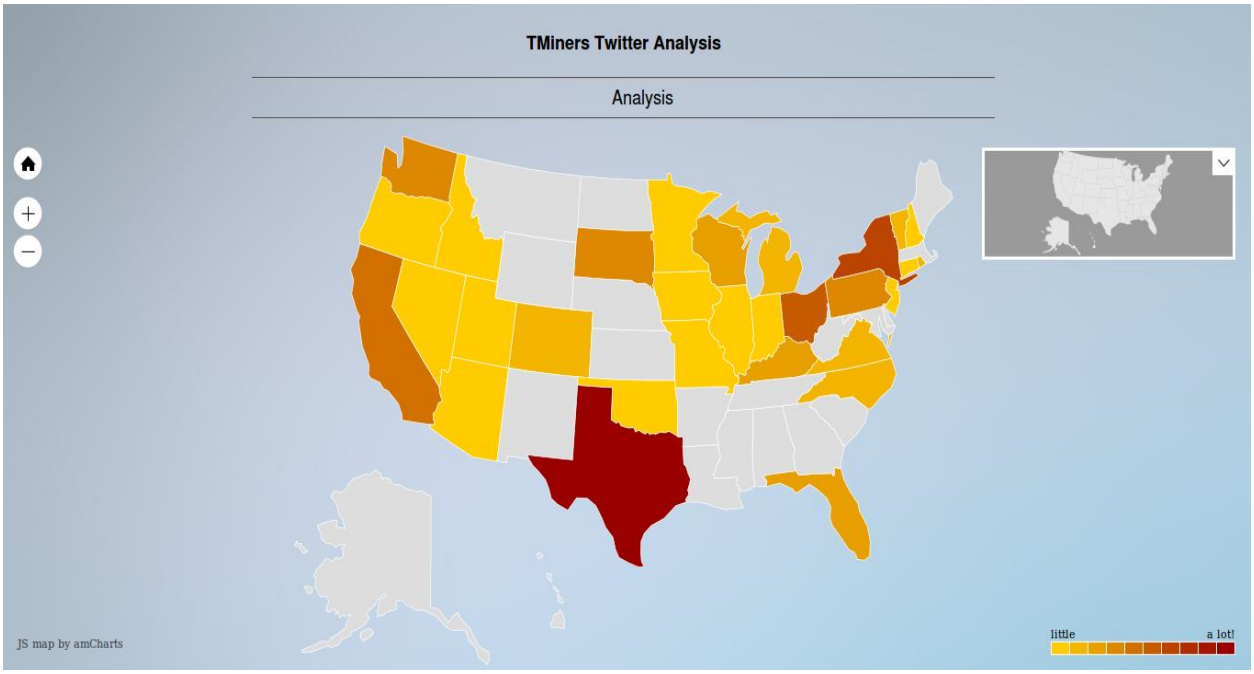
This is the front end of the application. User can enter keyword in the specified text box and click on "Generate Heat Map" or "Find Sentiment" based on his requirement. User can enter a maximum of 25 characters as the keyword. We have used this limitation since tweets are limited to only 140 characters.



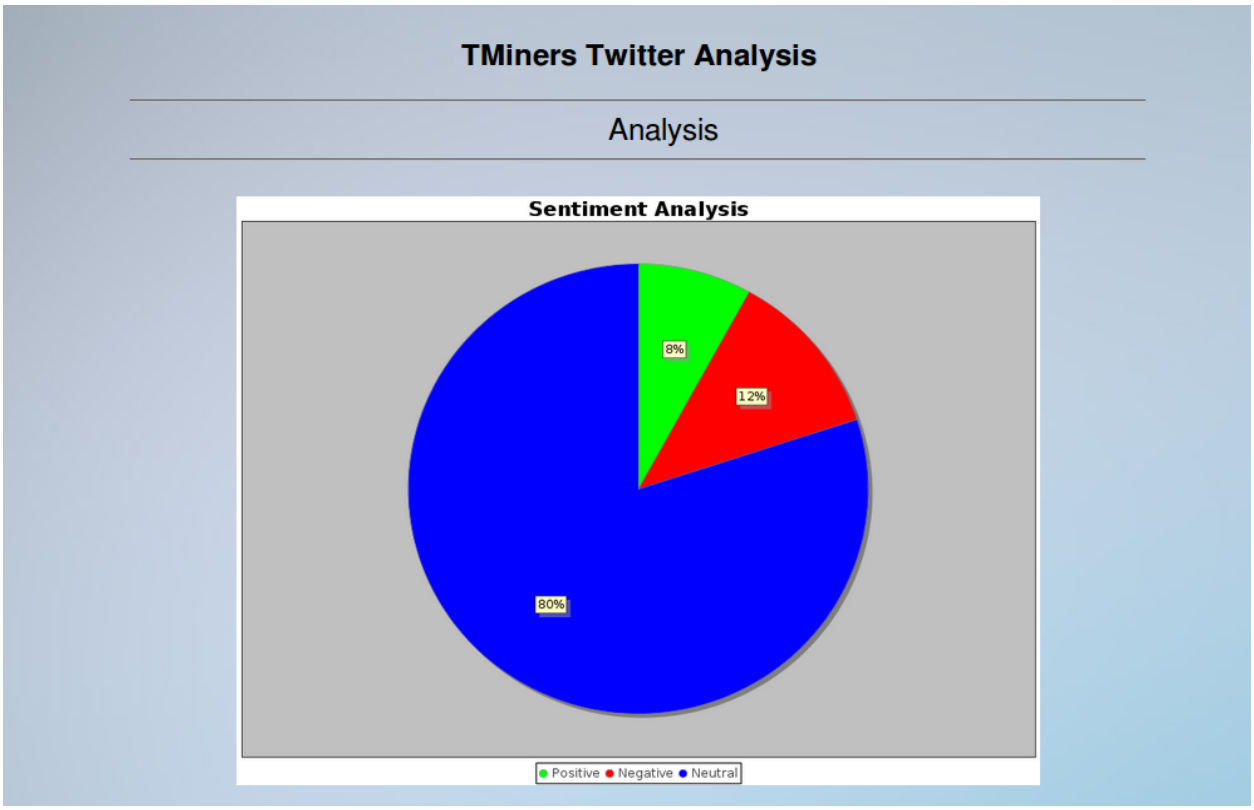
Keyword is the mandatory input in this application. If a user clicks on “Generate Heat Map” or “Find Sentiment” without entering keyword, an error message will be displayed.



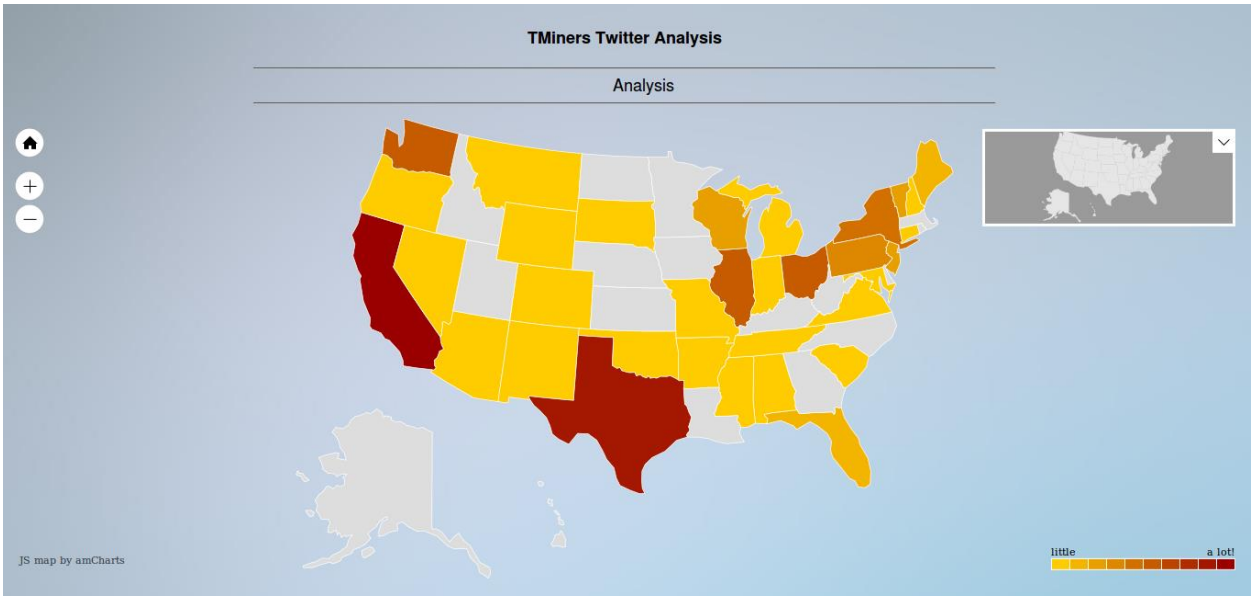
This map was generated when user entered keyword as “Chicago”. The color intensity indicates the density distribution of tweets for the word “Chicago”. We have used this word to validate our analysis as it shows that state “Illinois” has the highest density.



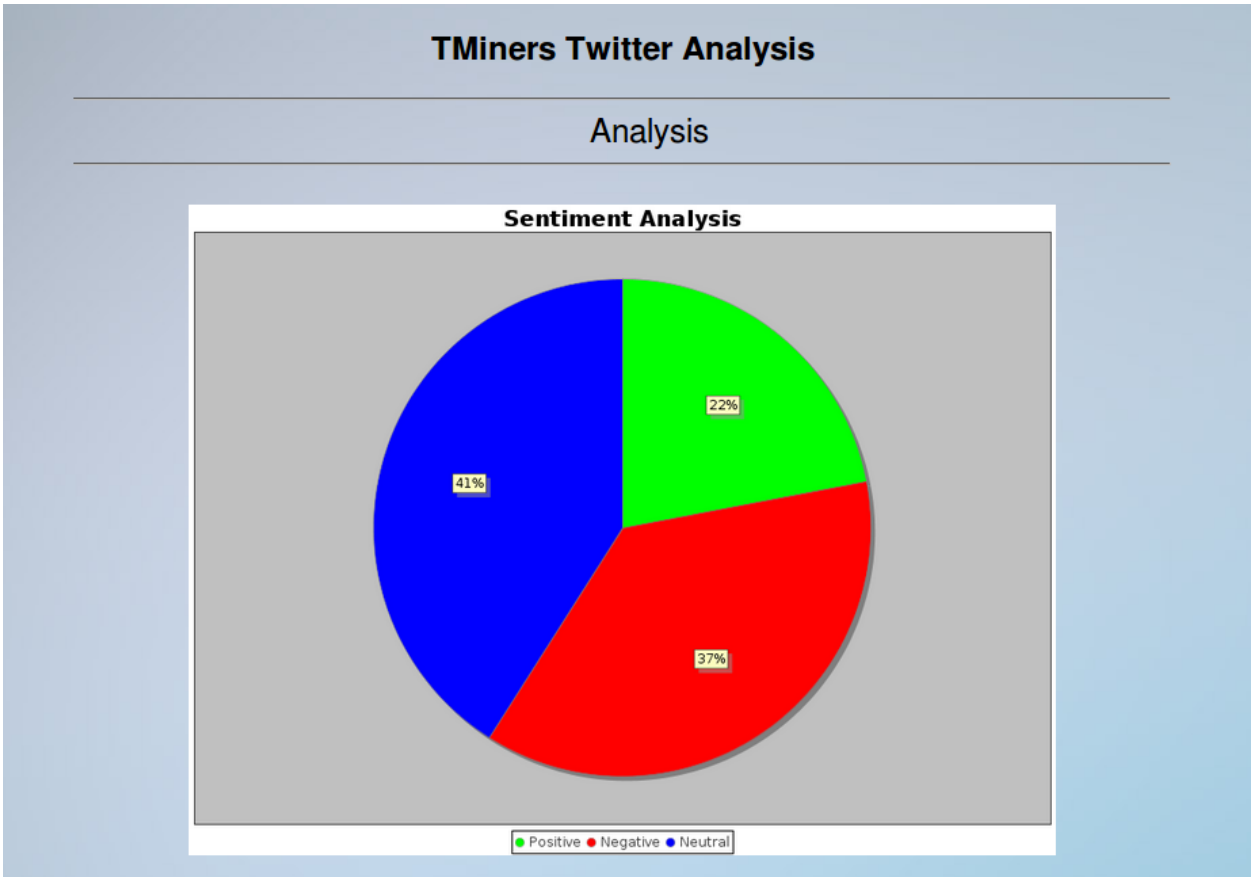
This map was generated when user entered keyword as “Obama”. Map indicates that Texas has the highest density of tweets followed by New York. States without any relevant tweets are represented by white color such as New Mexico, Montana, etc.



This sentiment analysis was generated when user entered keyword as “Obama”. The pie chart indicates that 8% of tweets have positive opinion, 12% of tweets have negative opinion and 80% of tweets have neutral opinion on Obama.



This map was generated when user entered keyword as “Starbucks”. Map indicates that Texas and California have the highest density of tweets followed by Washington, Illinois and Ohio. States without any relevant tweets are represented by white color such as North Dakota, Georgia, etc.



This sentiment analysis was generated when user entered keyword as “Starbucks”. The pie chart indicates that 22% of tweets have positive opinion, 37% of tweets have negative opinion and 41% of tweets have neutral opinion on Obama.

5. Conclusion

As the core functionalities of the application are implemented using Hadoop (MapReduce) and MongoDB, the application can be easily scaled to an enterprise level to perform analysis on huge amounts of data. The only bottle neck in the current application is the input data from twitter. By getting access to use Twitter Firehose, this application can be made more effective to use at enterprise level.

6. References

Twitter4j: http://twitter4j.org/en/code-examples.html

AmCharts: http://www.amcharts.com/demos/us-heat-map/

LingPipe: http://alias-i.com/lingpipe/

7. Learning from class

- IDS 561 course gave us introduction to big data technologies and Map Reduce framework
- We also got to know the implementation of these algorithms using the Map Reduce Framework along with the review of several Machine learning algorithms
- We learnt the evaluation of the models using different measures
- Different types of classifiers for sentiment analysis in-depth knowledge
- MongoDB – NoSQL database
- Working in Linux environment
- The project gave us a chance to work with the Hadoop file system and implementing several algorithms

8. Appendix:

Code Description:

Package	Classes	Description
edu.uic.ids561.Hadoop	Hadoop	This class is used to start/stop hadoop and run the jobs
edu.uic.ids561.SentimentAnalysis	Classifier	Classify the text into positive, negative or neutral
	SADriver	Driver class of MapReduce for sentiment analysis
	SAMapper	Mapper class of MapReduce for sentiment analysis
	SAReducer	Reducer class of MapReduce for sentiment analysis
edu.uic.ids561.Twitter	LoadMapData	Read the MapReduce output for density distribution of state location and convert to JSON format. After conversion to JSON format, store the data as JSON file
	LoadPieChart	Read the MapReduce output for sentiment analysis and convert to percentages and generate pie chart
	MongoDB	Build connection with MongoDB localhost and to insert Documents into Collection or drop DB/Collection or Read Documents in theCollection
	OAuth	OAuthorization of Twitter Stream data
	TwitterStream	Get connection with Twitter and fetch tweets. Methods to perform sentiment analysis and density distribution
edu.uic.ids561.WordCount	WCDriver	Driver class of MapReduce for density distribution of state location
	WCMapper	Mapper class of MapReduce for density distribution of state location
	WCReducer	Reducer class of MapReduce for density distribution of state location