INDIANA UNIVERISTY BLOOMINGTON

TWITTER DATASET ANALYSIS AND MODELING

Analysis Track (Track 1)

B669 Management, Access and Use of Big and Complex Data

PROJECT REPORT

Submitted by

Nihar Khetan
Masters Candidate in Computer Science
School of Informatics and Computing
Indiana University
nkhetan@indiana.edu (#0003402419)

Under the Guidance of:

Professor Beth Plale
Director, Data to Insight Center
Managing Director, PTI
Professor, School of
Informatics and Computing
Indiana University

Yuan Luo
Ph.D. Candidate in Computer Science
School of Informatics and Computing
Indiana University

ACKNOWLEDGMENT

This project would not have been possible without Twitter Dataset from the works of Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, Kevin Chen-Chuan Chang: Towards Social user profiling: unified an discriminative influence model for inferring home locations. KDD 2012:1023-1031

I would like to thank **Professor Beth Plale,** for giving us an opportunity to do this project and encouraging us throughout.

I am also grateful to our Associate Instructor **Yuan Luo**, for the constant help and support during the course of the project. Thanks for being proactive on the discussion forum which made solving doubts easier and convenient. Thanks again for designing the project wonderfully which helped me learn not only MongoDb but also taught me how to handle such large volumes of data.

Nihar Khetan #0003402419

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Problem Description

Analysis Track 1

The objective of this track is to validate, label, analyze and visualize a Twitter dataset.

Tasks:

a) Reformat dataset. The raw txt file of user profiles is encoded in ISO-8859-1, a format MongoDB does not accept. The txt file will need conversion to UTF-8 format.
b) Ingest the reformatted data into MongoDB.
c) Validate user locations and get geo-coordinates. The Twitter user profile allows user to input of arbitrary text to indicate their location. The user locations might not be recognizable. Use Google geocoding API to validate user locations, extract valid Latitude/Longitude of the user locations.
d) Store user location coordinates into MongoDB.
e) Visualize selected user locations using Google Maps.

1.2 Data Cleaning

User profiles in the twitter dataset contain the following data.

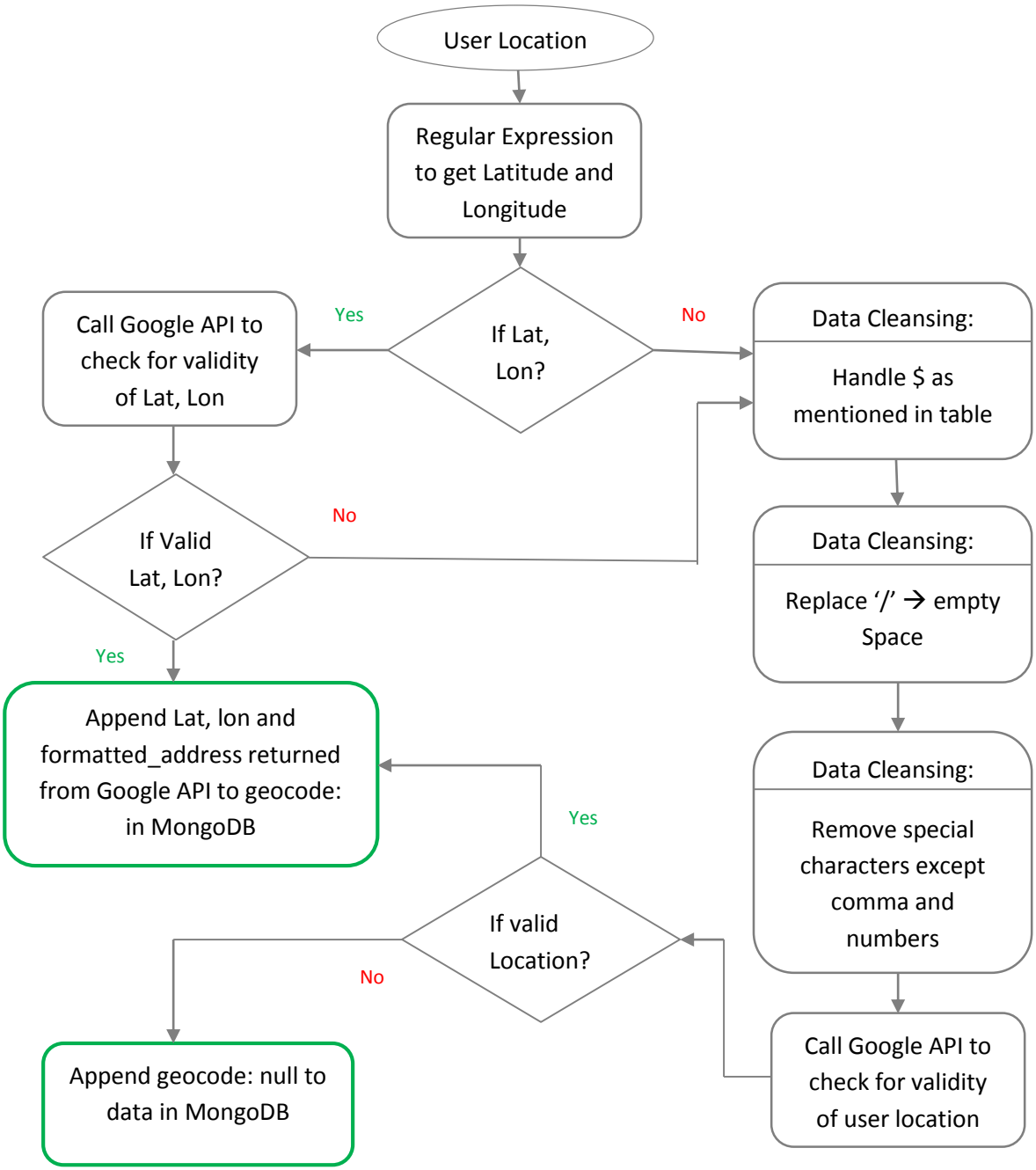
USER ID, USER NAME, FRIEND COUNT, FOLLOWER COUNT, STATUS COUNT, FAVORITE COUNT, ACCOUNT AGE, **USER LOCATION**

User Location is ideally the geolocation from where the user is tweeting from. User Location data present in the dataset requires lot of data cleaning to get proper geolocations. On Initial run **without any data processing 15 user locations could not be processed to have geocode: null.** Example: Toronto XYZ ONT

However various preliminary data cleaning strategies were employed to clean User Location data which are listed below.

Strategy	***Description***	***Example***	
		Before	***After***
Remove all special characters	All special characters to be removed **except comma and numbers** because addresses might be comma separated	^Atlanta,*& Georgia *##@!!	Atlanta, Georgia
Forward Slash	Convert all forward slashes to empty spaces. Lot of addresses have delimiters as forward slashes (/). Converting them to empty space helped Google API to recognize such addresses	LA/Downey CA Austin/Arlington/TX	LA Downey CA Austin Arlington TX
Handle $	$ symbol is handles very carefully because there were lot of addresses where converting $ → S made it a valid address. Thus if there are characters (alphabets) on left of dollar or on right of dollar symbol convert it to 'S', otherwise treat it as special character and remove it	La$ Vega$ **** EA$T OAKLAND $ Vega$ $$ US(!)	LaS VegaS EAST OAKLAND VegaS US

Some user locations were directly given as **Latitude/Longitude** which were both valid as well as invalid coordinates. So more processing was done to get accurate and consistent data. Flow diagram below explains the process.



Examples:

Earth..u ^&&*^*$
Query: earth+u *(NOT VALID)*
{“geocode” : null}

Vega$, NYC, OHiiiiO
Query: VegaS,+NYC,+OHiiio *(VALID)*
{“geocode” : {
“formatted_address” : “Las Vegas, NV, USA”,
“location” : {
“lat” : “36.1699412”,
“lon”: “-115.1398296”
}}}

iphone:28.113894,82.378614Query: VegaS,+NYC,+OHiiio
Reverse Query: latlng=28.113894,82.378614 *(VALID)*
{“geocode” : {
“formatted_address” : “Chillikot Road, Hapur 22412, Nepal”,
“location” : {
“lat” : “28.113894”,
“lon”: “82.378614”
}}}

ÜT: 6.900301,107.602353
Reverse Query: latlng=6.900301,107.602353 *(NOT VALID)*
{“geocode” : null}

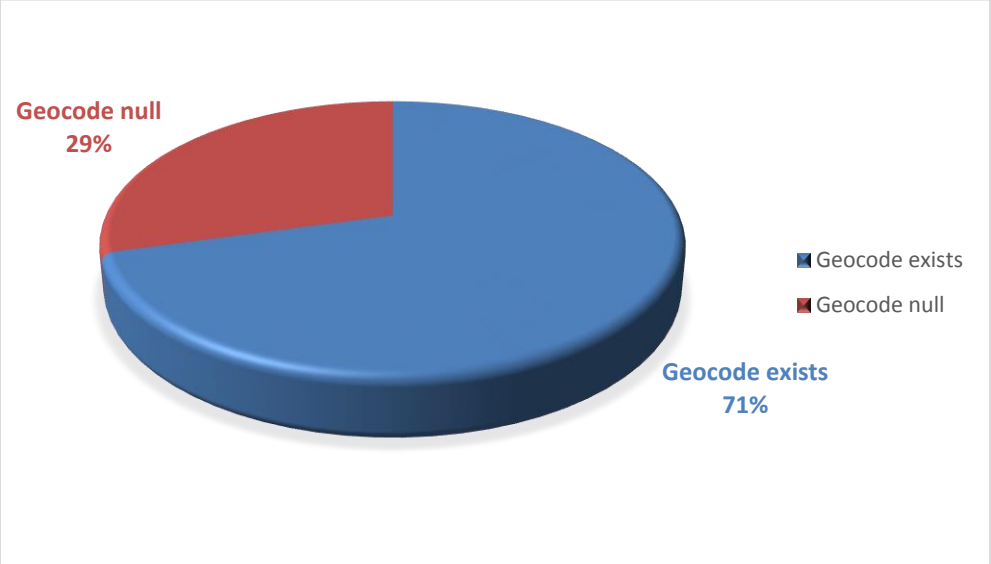
Chapter 2

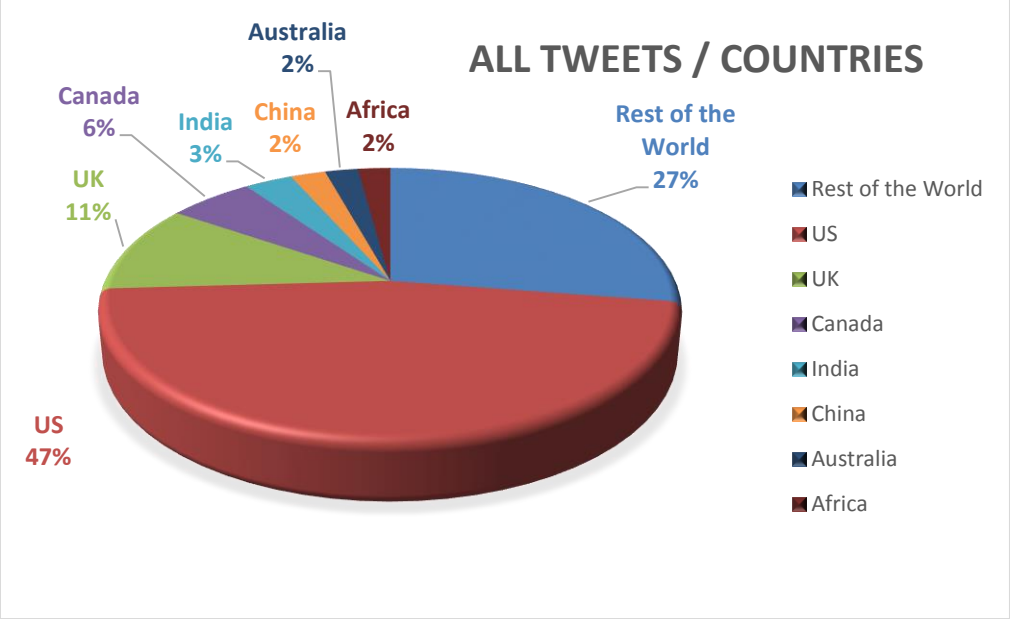
GEOCODING

2.1 Geocoding Statistics

Geocoded	*Not Geocoded*	***Total***
7069	**2931***	10000

***Out of 2931 not geocoded records 1390 have user_location : “” <empty>**





2.2 Suggestions to solve "geocode": "null"

Most of the geocodes which cannot be mapped does not make any sense and have random text.

There are 1390 records with no user location mentioned.

User Locations which can be potentially mapped have 3 problems:

I. Multiple spelling mistakes by users
   a. Example: Mas Megas
II. Coordinates which map to nowhere
   a. Example: ÜT: 6.900301,107.602353
III. Places without spaces
   a. Example: BUDAPESTGERMANY

I and III can be corrected by using following techniques:

- By doing natural language processing to catch *spelling mistakes* and use *spell check algorithms* to find words of closest match. Example: Edit distance (Levenshtein distance). Weighted edit distance, n-gram overlap.
- Multiple attempts can be made to generate the closest match and google api can be called again and again to find if valid match is found.

It can be fixed by first checking if Latitude and Longitude is semantically correct. If it is correct then *closest matching location* in terms of distance can be mapped to the given Latitude and Longitude.

Chapter 3

QUERY CRITERIA AND PERFORMANCE

3.1 Alternate Solutions to query criteria

After ingesting Twitter Users Data in MongoDB multiple techniques can be used a alternate query criteria:

a. Get all empty user locations by query as :
    - db.users.find({‘user_location’: {$lte: “”}})
    - Do a bulk insert for *geocode: null* for such locations.
    - Handle non empty user locations by normal query criteria as given in project.

b. Data can be accessed by querying for each location. For example USA, Australia, China; and then a query can be formed with *not* of locations initially queried for.

c. User locations have text as well as coordinates, so regular expressions can be used to handle both separately as our goal is to; text -> extract coordinates and coordinated-> extract user location using Google Geocoding API.

d. Data can be processed and cleaned as it is already done in Chapter 1.2

3.2 How to improve performance for large datasets?

There are multiple ways to improve performance in case of large datasets:

a. **Parallel Processing:** two threads can be used as T1 and T2 working simultaneously. Task of thread T1: get Five Hundred User locations from MongoDB each time and process/clean them. Task of thread T2: get records from T1 and call Google Geocoding API on them. Thus some efficiency is achieved.
    - Google geocoding can further be expedited by having a paid customer subscription. Ten records are processed per second instead of five.

b. **Indexing:** Indexing on user_location field and then using $or as query criteria
    - Virtually splitting up the locations by country
    - Querying on two countries at a time
    - Indexing speeds up the query time in case of large dataset

c. **Bulk Modify/Insert:** Along with Parallel Processing and indexing this technique also speeds up the writes to MongoDB.
    - Instead of updating one geocode at a time, process multiple geocodes (say five hundred) and do a Bulk Insert in a new collection or Bulk Modify (challenging) in the same collection.

d. **Using Geospatial Query Operators:** [1] Geospatial query operators like $geoWithin can be used with predefine latitudes and longitudes. However this method can only be used to get user locations which are already present as coordinates. Thus with very large dataset user locations can be imagined to be spread all across the world. Thus world can be split in blocks (by latitudes and longitudes). Each block is handles separately for geocoding.

e. **Map Reduce** [2] can be used to make query process faster in case of large data set.

Chapter 4

RUNNING VISUALIZATION APP

A Web Application was developed using SAPUI5 framework which can be rendered on **any device** (mobile or desktop).

Follow the steps below to deploy and run the app on local server:

1. Install jdk on your machine: Link: http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Install eclipse juno on your machine, Link: http://www.eclipse.org/downloads/packages/release/Juno/SR2
3. Launch eclipse and install Apache Tomcat 7 via eclipse.
4. Install SAPUI5 Plugin to eclipse:
    a. Eclipse → Help → Install New Software.
    b. Paste this link: https://tools.hana.ondemand.com/juno
    c. Install SAPUI5 components and restart eclipse.
5. Now eclipse is ready to run the project.
6. Import project File → Import → Import existing projects in workspace.
7. To Run: Right click on project → Run on server → Select Apache Tomcat7 → Run.
8. Open url on your browser as : http://localhost:8080/TwitterVisualization/
9. You can emulate the app environment to see how it runs on mobile devices on chrome browser:
    a. Press F12.
    b. In the console between search symbol and ‘elements’ on header bar there will be a mobile icon. Press that.
    c. Select the device you want to emulate from dropdown on the left.
    d. App looks as if it is running on that device.

***** Tweet Visualization takes some time to load as Google API maps to latitudes and longitudes and renders it to the screen.***

I REFERENCES:

Twitter Dataset

- https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012
- Rui Li, Shengjie Wang, Kevin Chen-Chuan Chang: Multiple Location Profiling for Users and Relationships from Social Network and Content PVLDB 5(11):1603-1614, 2012
- Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, Kevin Chen-Chuan Chang: Towards social user profiling: unified and discriminative influence model for inferring home locations. KDD 2012:1023-1031

MongoDB

- http://www.mongodb.org/
- http://docs.mongodb.org/manual/tutorial/query-documents/

Geocoding

- https://developers.google.com/maps/documentation/geocoding/

Visualization

- https://developers.google.com/chart/interactive/docs/gallery/map
- SAPUI5 (Framework Used for Visualization)
    - https://sapui5.netweaver.ondemand.com/sdk/

Others

- Ppt Information Retrieval techniques: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCAQFjAA&url=http%3A%2F%2Fweb.stanford.edu%2Fclass%2Fcs276%2Fhandouts%2Flecture3-tolerant-retrieval.ppt&ei=nbdfVOWIN4SayQTyxIG4Cg&usg=AFQjCNFVbcWGCADiGD1VmeMck1G-F8kcfw&sig2=JLXdU-4RJhw9yLCpVmJqyg
- [1] http://docs.mongodb.org/manual/reference/operator/query-geospatial/
- [2] http://docs.mongodb.org/manual/core/map-reduce/

II Images

```
> db.users7.find({"geocode" :{$exists:true, $lte:null}}).count()
2931
> db.users7.find({"geocode" :{$exists:true, $ne:null}}).count()
7069
>
```

Image 1: Count of User Locations which could be mapped and which could not be mapped

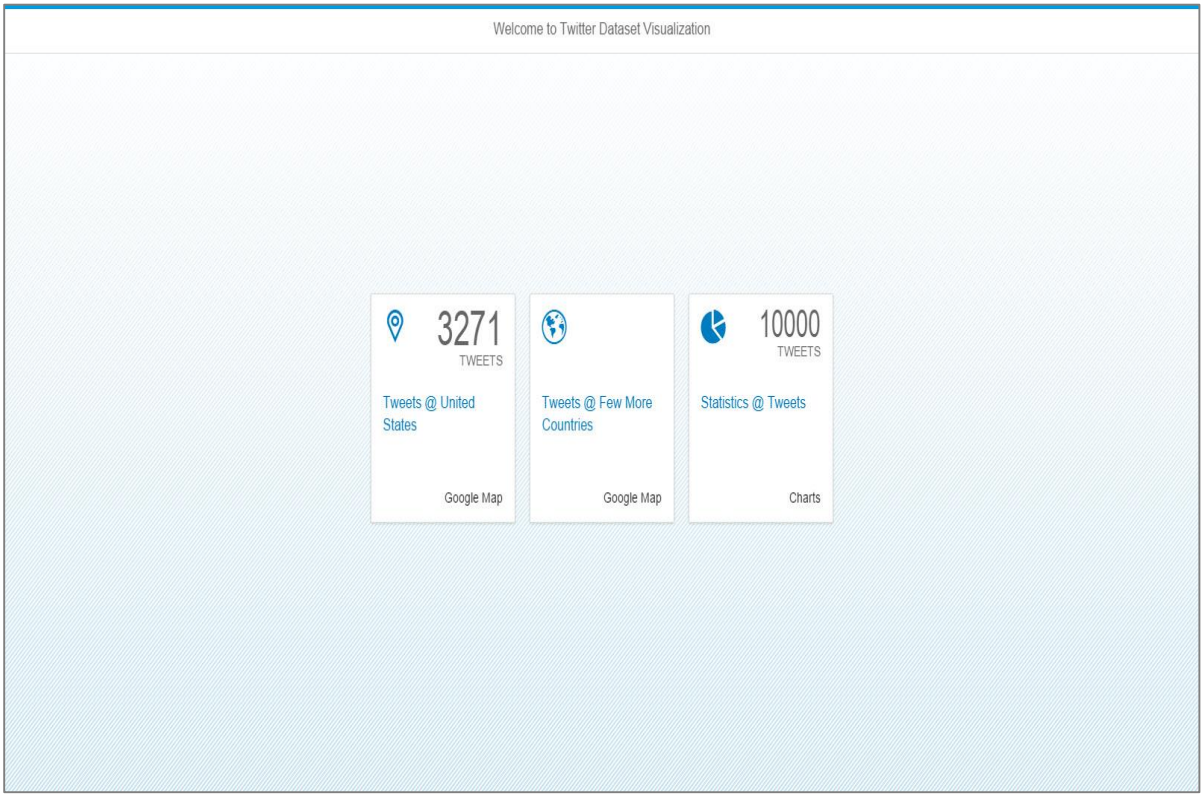


Image 2: Home Screen for Data Visualization App

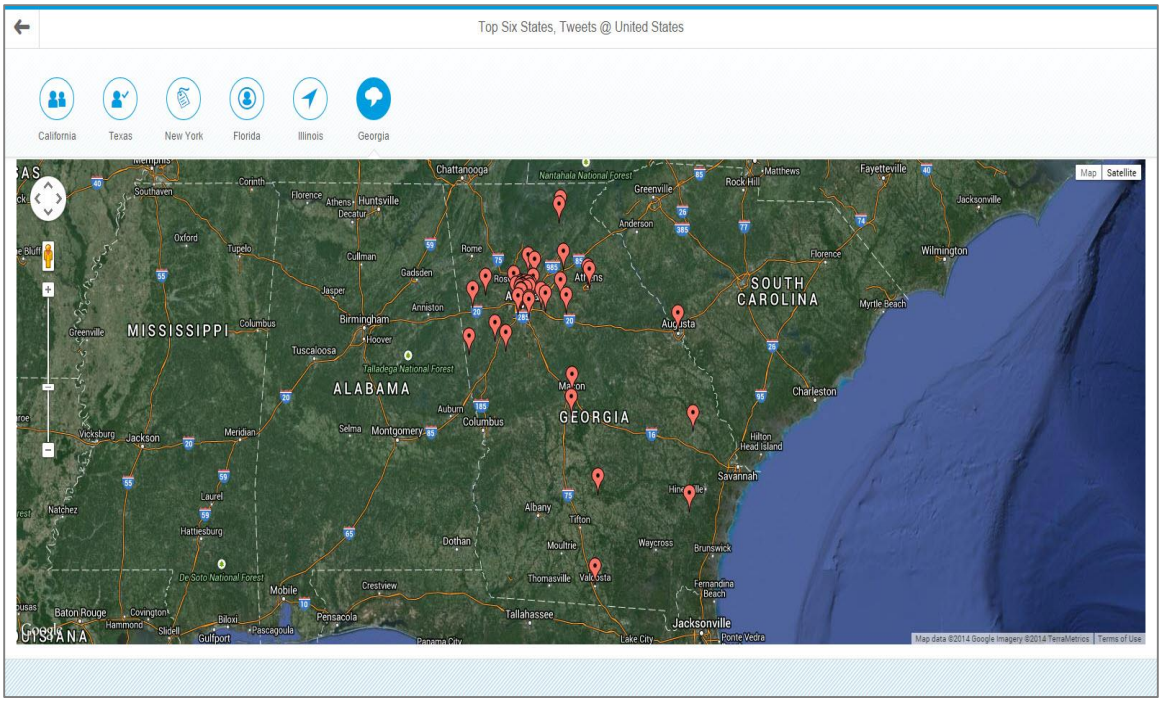


Image 3: Screen 1 Visualizing top six most tweeting states in the United States

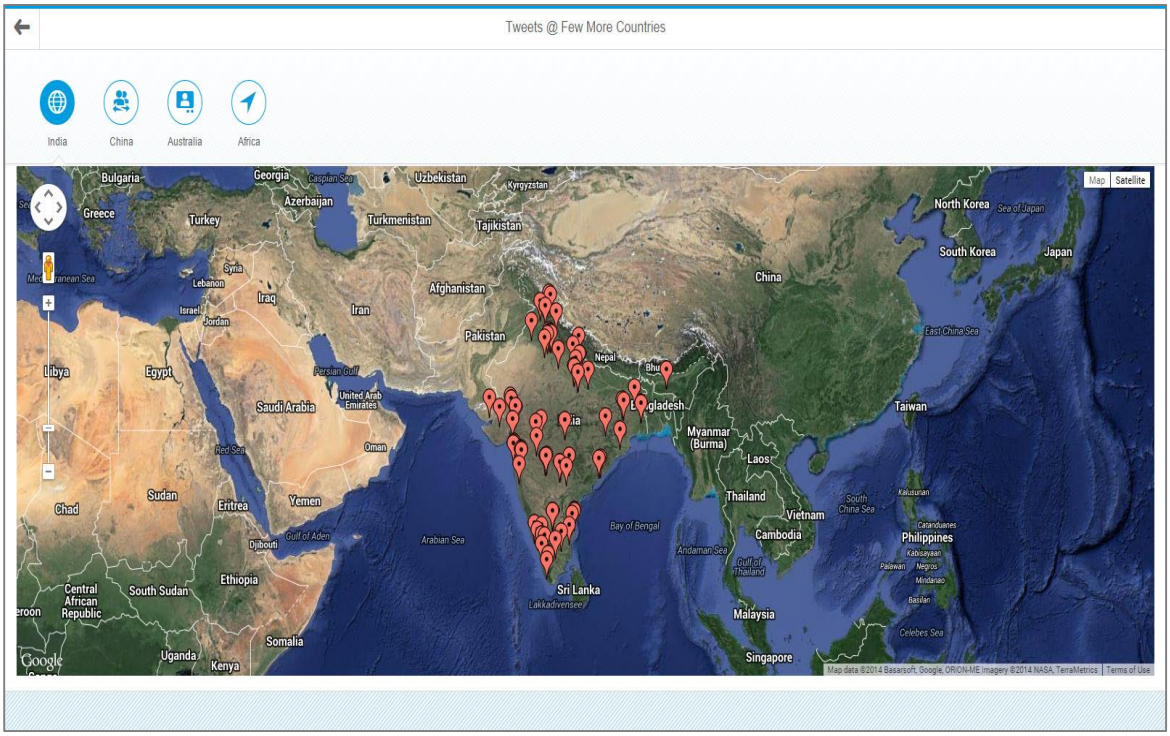


Image 4: Screen 2 Visualizing few other tweeting countries in the world

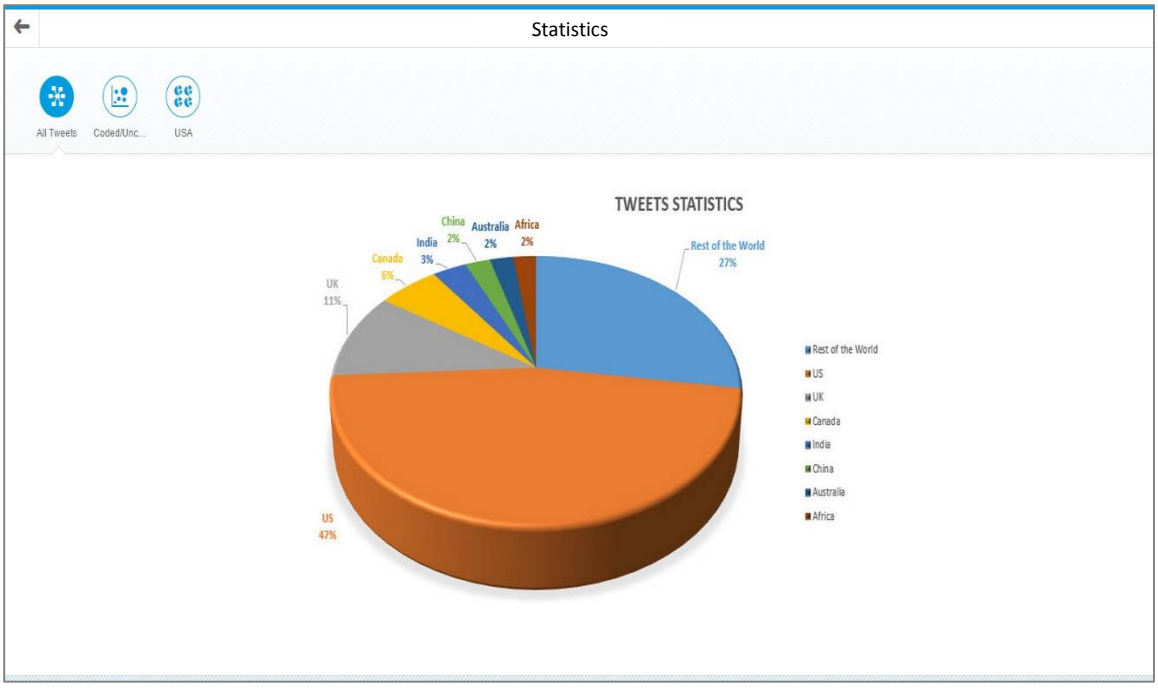


Image 5: Pie chart: Distribution of all geocoded user locations

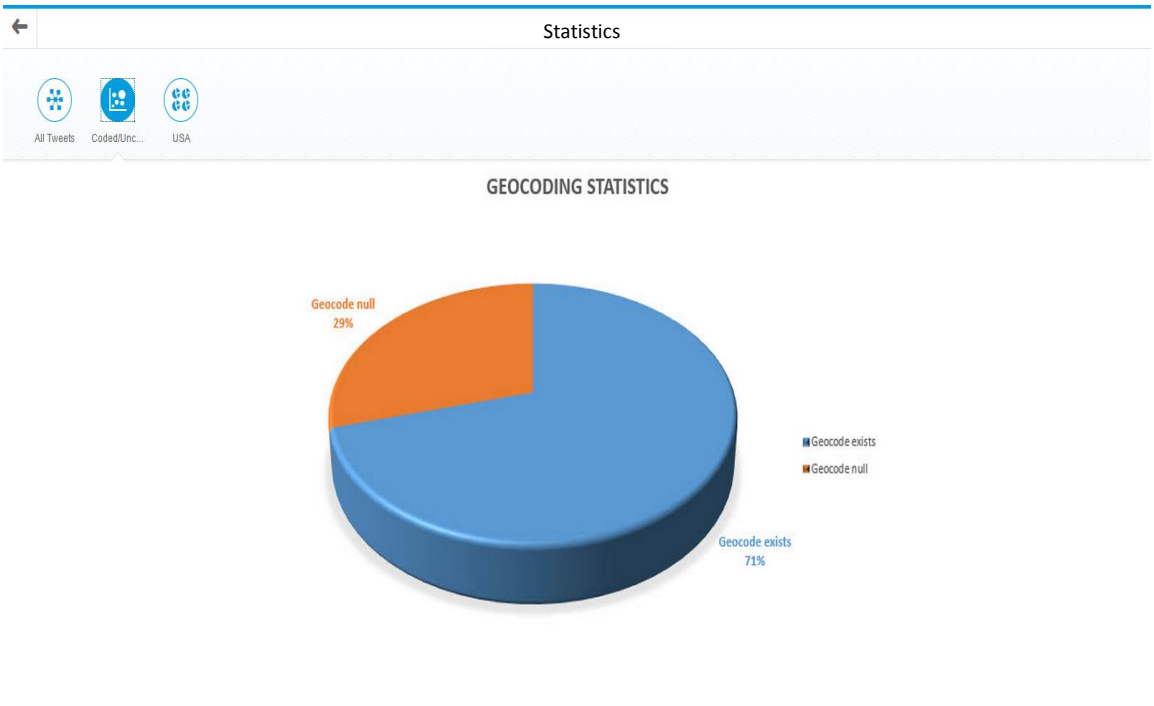


Image 6: Pie chart: Distribution of geocoded / non geocoded

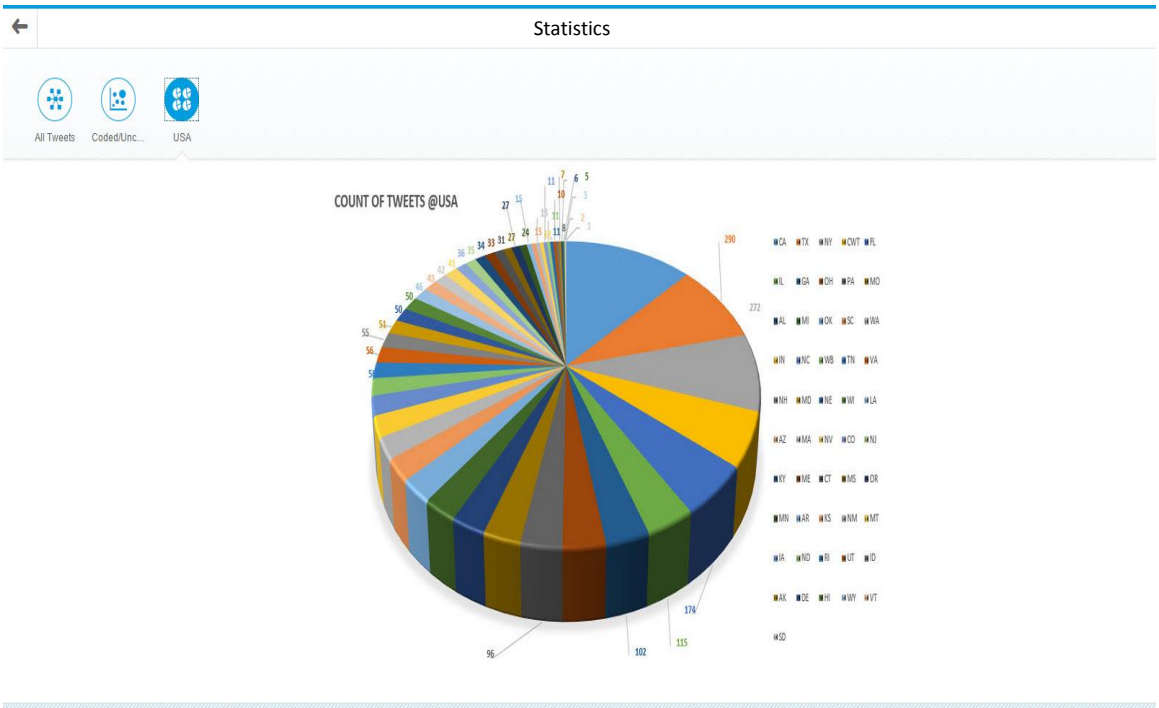


Image 7: Pie chart: Distribution of all American tweeting states

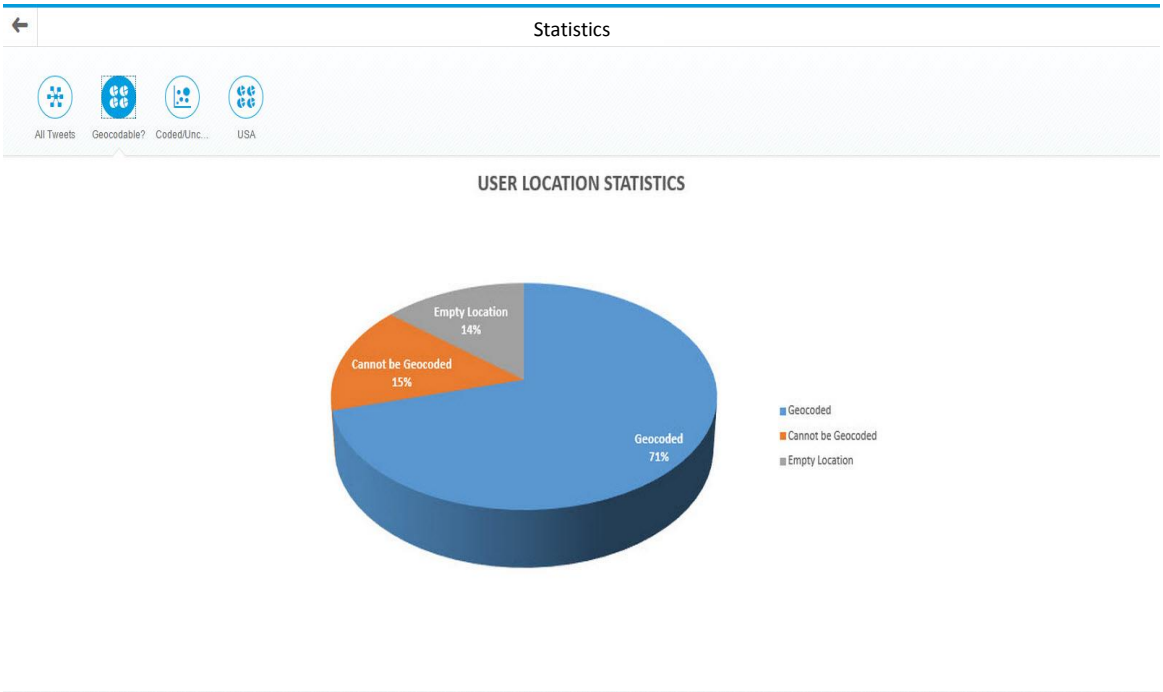


Image 8: Pie Chart: Distribution of User Locations (Valid , Invalid (null, junk))

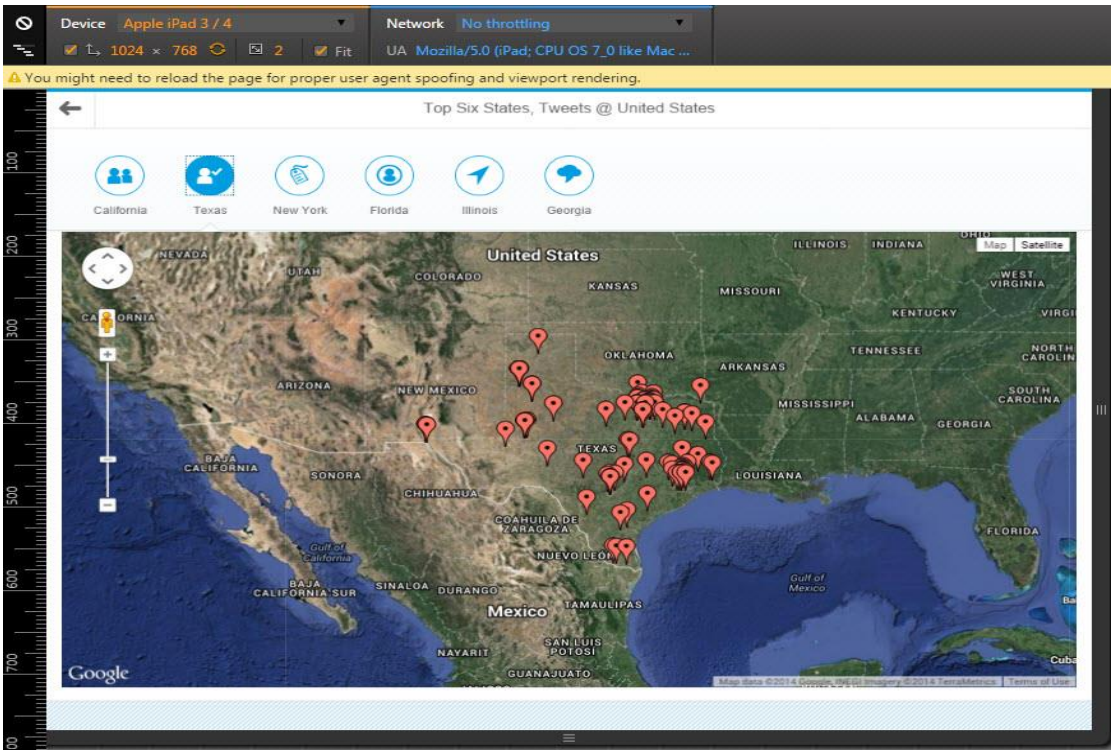


Image 9: App running on **APPLE IPAD MINI**

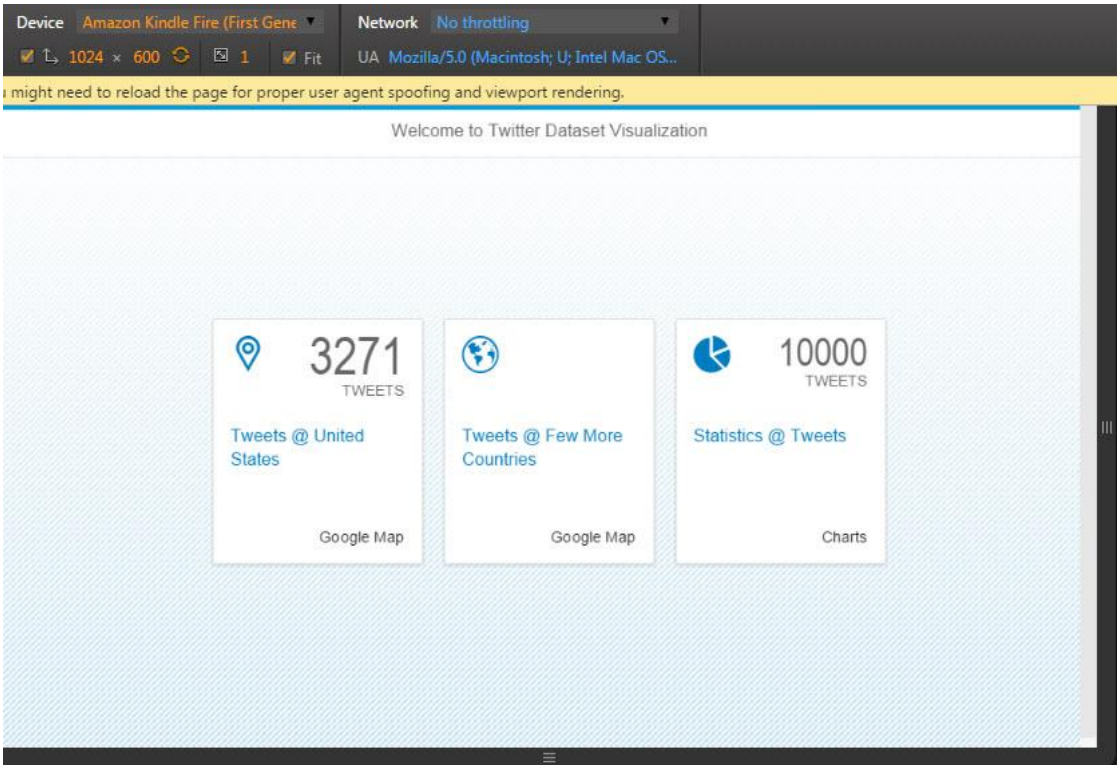


Image 10: App running on **AMAZON KINDLE FIRE (first generation)**

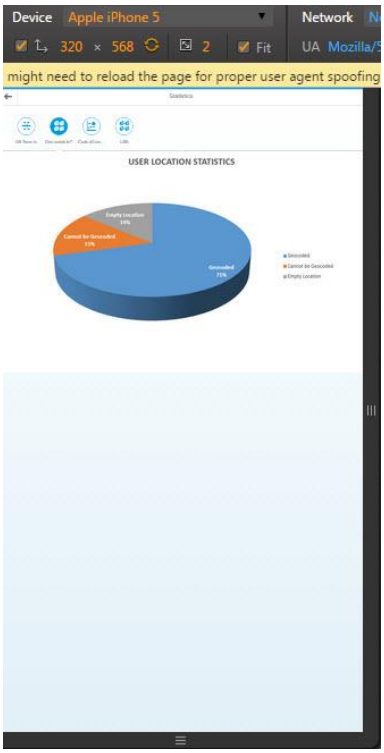


Image 11: App running on **APPLE IPHONE 5**