Nibu Jacob
FA14-BL-INFO-I590-34511
School of Informatics and Computing
Indiana University

Twitter Dataset Analysis and Modeling

The purpose of this project is to validate location data in a given set of Twitter data and visualize it on Google map using Google APIs. The Twitter data(1) used is a subset of a Twitter data set used by Rui L., Shengjie W., Hongbo D., Rui W., and Kevin C. C.(2)

Data preparation and import

A VirtualBox of Ubuntu with MongoDB installed was used for this project. A small Java project code was provided and this was deployed using Ant.

The Twitter data was in a text file encoded in ISO-8859-1 with tab-separated values and no header line. Since MongoDB uses BSON as the data storage and network transfer format for JSON documents that it stores(3) and BSON can only be encoded in UTF-8(4), the text file encoding was changed to UTF-8 using the given shell script:

```
$ ./bin/reformat.sh ./data/users_10000.txt ./data/users.txt
```

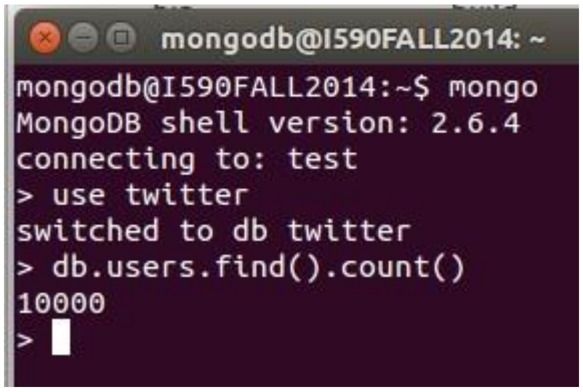
Since the text file does not have a header line, headings were added manually by typing in tab-separated headings in users.txt:

```
user_id user_name friend_count follower_count status_count favorite_count account_age user_location
```

The data was then imported into MongoDB (twitter database and users collection) using the given shell script:

```
$ ./bin/import_mangodb.sh twitter users tsv ./data/users.txt
```

Verified successful import of the data by querying in a mongo shell(5):



```
mongodb@I590FALL2014:~$ mongo
MongoDB shell version: 2.6.4
connecting to: test
> use twitter
switched to db twitter
> db.users.find().count()
10000
>
```

Data validation

In order to visualize the data on Google map, the user documents must contain geocodes of user locations. Each user document contains a user_location field that has arbitrary string values. Twitter lets its users to input any text as user location. The field is nullable and the string input by the users need not necessarily be a location nor parseable(6). Therefore, the values in this field must be validated as locations and if so the corresponding geocodes must be found and added as additional fields in the document.

Validation and geocoding of user location data was done using Google Geocoding API(7). The given shell script and query JSON document were used to run the geocoding process:

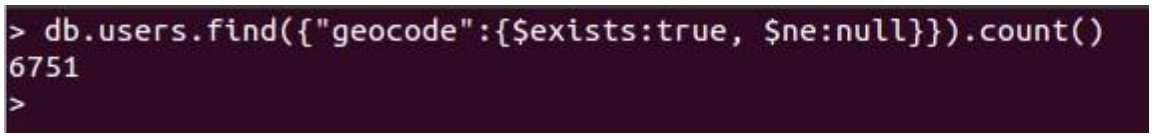
```
$ ./bin/QueryAndUpdate.sh ./config/config.properties twitter users ./input/query.json ./log/query.log
```

Since the free Google Geocoding API has a usage limit of 2,500 requests per 24 hours(7) the above command was run over 4 days to complete the 10,000 documents.

Google Geocoding API accepts only strings that are URL encoded. A valid URL cannot contain special characters and spaces (14). The given code takes care of URL encoding and substitutes spaces with + sign.

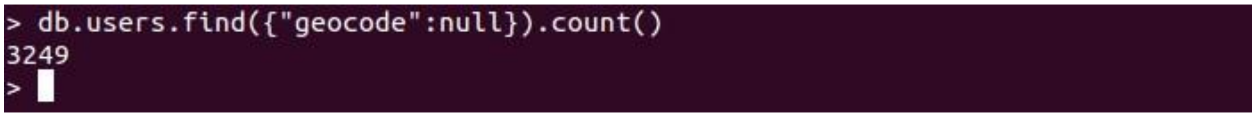
Validation of locations

To check how many documents were successfully validated and geocoded, the following query was run in mongo shell:



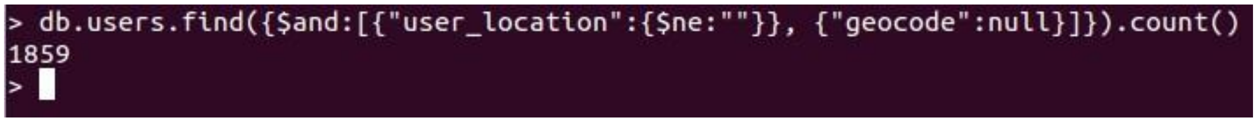
```
> db.users.find({"geocode":{$exists:true, $ne:null}}).count()
6751
>
```

The remaining documents were found by the following query:



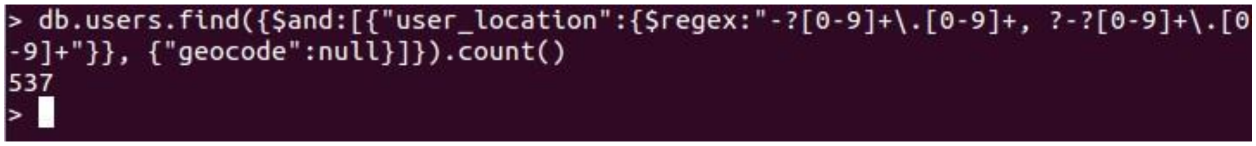
```
> db.users.find({"geocode":null}).count()
3249
>
```

To resolve this, the user_location field was explored further to see what could be done to improve validation and geocoding. The following query revealed that, of the 3249 documents without geocodes, 1859 documents have their user_location field empty:



```
> db.users.find({$and:[{"user_location":{$ne:""}}, {"geocode":null}]}).count()
1859
>
```

Nothing could be done about these documents. Examination of the remaining documents revealed that some of them contain longitude and latitude data in their user_location field. The number of such documents were found by the following query using regex operator(8):



```
> db.users.find({$and:[{"user_location":{$regex:"-?[0-9]+\.[0-9]+, ?-?[0-9]+\.[0
-9]+"}}, {"geocode":null}]}).count()
537
>
```

Google Geocoding API offers a reverse geocoding facility(7) using which the user_location field can be populated for these documents. An example usage of the API is:

```
https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452
```

The rest of the documents can be assumed to contain irrelevant text. Although it is possible to extract location data using geocoding algorithms on Twitter content and metadata(9) a large percentage of users are expected to have either non-sensical or missing data in their user_location field(10).

Query criteria and performance

The sample query criteria queries the database to fetch all documents that do not contain the "geocode" field. An alternative way is to exclude documents that have geocodes in their user_location field:

{$and:[{"geocode":{$exists:false}}, {"user_location":{$ne:{$regex:"-?[0-9]+\.[0-9]+, ?-?[0-9]+\.[0-9]+"}}}]}

Another option is to exclude documents that have user_location field empty:

{$and:[{"geocode":{$exists:false}}, {"user_location":{$ne:""}}]}

Now that some of the user documents are known to contain empty or geocoded user_location field, the best query criteria would be to exclude documents of both types:

```
{$and:[{“geocode”:{$exists:false}}, {“user_location”:{$ne:””}}, {“user_location”:{$ne:{$regex:”-?[0-9]+\.[0-9]+, ?-?[0-9]+\.[0-9]+”}}}]}
```

If the collection is very big, the query should limit the number of documents fetched at one time. Since Google Geocoding API has a daily limit of 2,500 requests it is a good choice to limit the documents retrieved to 2,500. This can be done using the limit() function(11):

```
> db.users.find({$and:[{“geocode”:{$exists:false}}, {“user_location”:{$ne:””}}, {“user_location”:{$ne:{$regex:”-?[0-9]+\.[0-9]+, ?-?[0-9]+\.[0-9]+”}}}]}).limit(2500)
```

The validated data was dumped from the database for turning in using the mongodump tool(12):

```
$ mongodump –d twitter –c users
```

Visualization

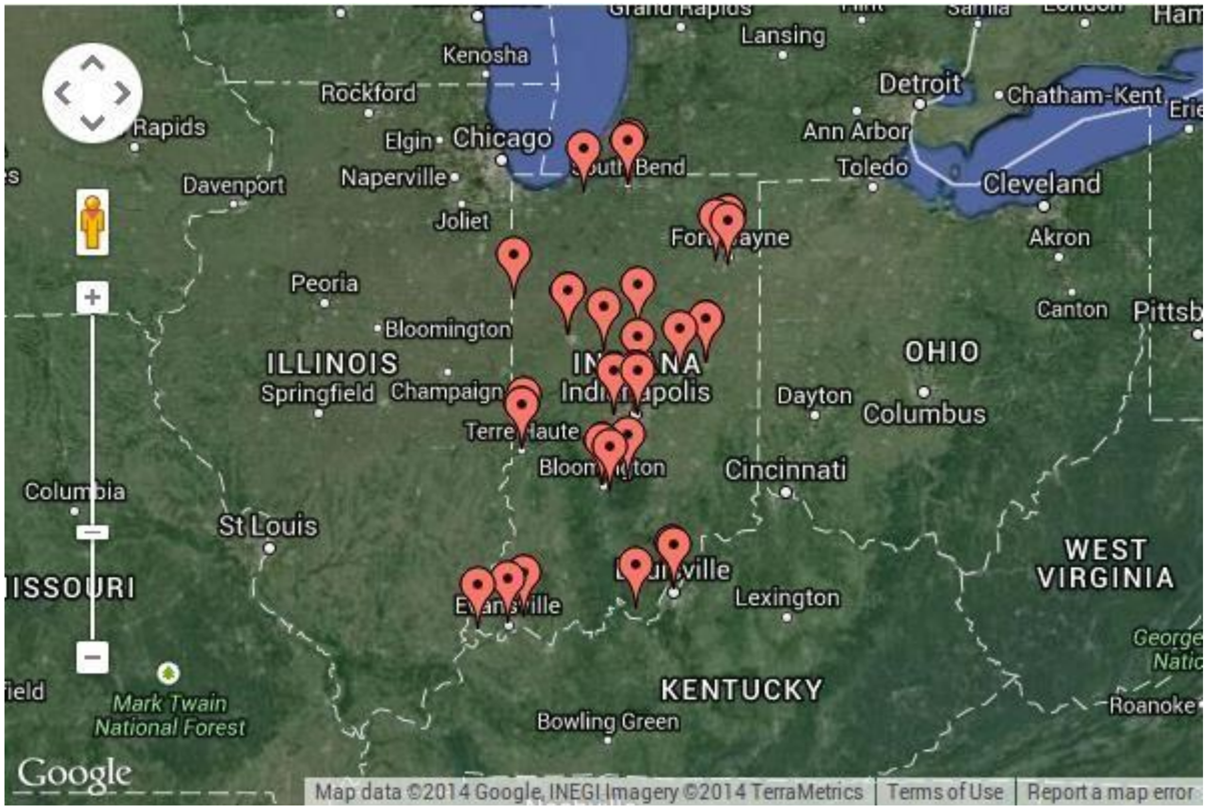
A subset of the data containing both “IN” and “USA” in formatted_address field (67 records) was exported from the MongoDB database to a CSV file using mongoexport tool:

```
$ mongoexport -d twitter -c users -q "{\"geocode\": {\$exists: true,\$ne:null}, \"geocode.formatted_address\":{\$regex: \"USA\"},\"geocode.formatted_address\":{ \$regex: \"IN\" }}" --csv  --fields geocode.formatted_address,user_name -o ./data/exportData.csv
```

Google Map Chart was used to visualize the data. The data was formatted as required by Google Map Chart(13):

```
$ awk '{ printf("[ %s ],\n", $l);}' ./data/exportData.csv
```

The output of the above command was copy-pasted into the example HTML code given in Google Map Chart page(13). The HTML file was then modified to display the project details. The output map is:



Conclusion

The given Twitter data was formatted for import into MongoDB and then validated and geocoded using Google Geocoding API. The documents that were left without geocodes were explored and it was found that some of them had geocodes in the location field. A subset of the validated data was used to visualize on Google map using Google Map Chart that uses Google Maps JavaScript API.

References

1. 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
2. https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012
3. http://docs.mongodb.org/meta-driver/latest/legacy/bson/
4. http://api.mongodb.org/erlang/bson/bson.html
5. http://docs.mongodb.org/manual/tutorial/query-documents/

6. https://dev.twitter.com/overview/api/users
7. https://developers.google.com/maps/documentation/geocoding/
8. http://docs.mongodb.org/manual/reference/operator/query/regex/
9. http://journals.uic.edu/ojs/index.php/fm/article/view/4366/3654
10. http://faculty.cse.tamu.edu/caverlee/pubs/cheng13tist.pdf
11. http://docs.mongodb.org/manual/reference/method/cursor.limit/
12. http://docs.mongodb.org/manual/reference/program/mongodump/
13. https://developers.google.com/chart/interactive/docs/gallery/map
14. https://developers.google.com/maps/documentation/webservices/