Algorithms for Geospatial Data Matching

An algorithm for error detection and correction of spatial data

Group Members

Surajnath Sidh	U101114FCS146
Shubham Singh	U101114FCS192
Mehak Bhatia	U101114FCS193
Kartik Bhadada	U101114FCS078

Project Mentor: Dr. Prosenjit Gupta

Rationale and Objectives of the work

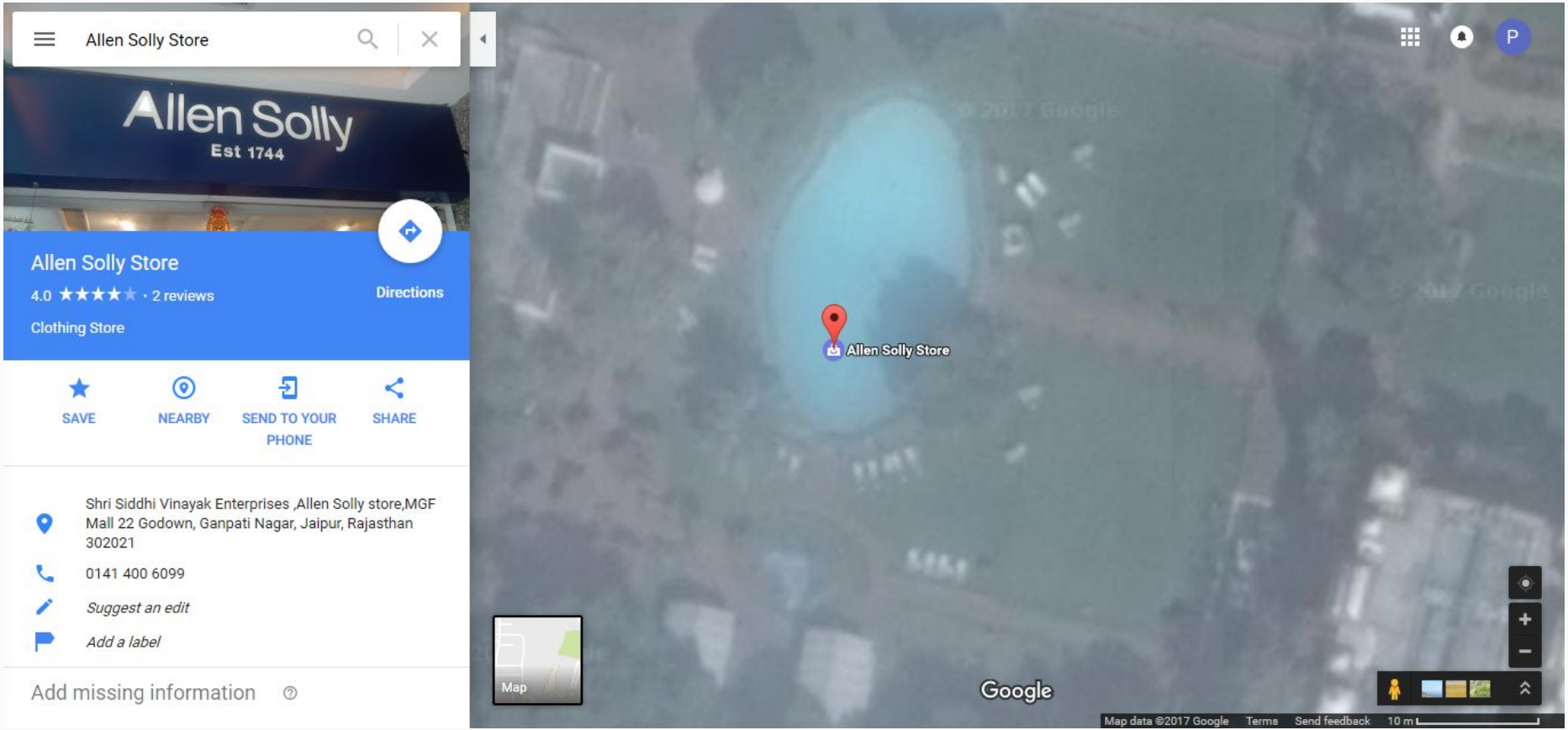
Objectives

- To bring attention towards data integration and data matching.
- To increase focus towards correcting erroneous data.
- To create an efficient algorithm for dynamic error detection, correction and updating the location of different places(hotels, restaurants etc) using retrieved or crowdsourced data.

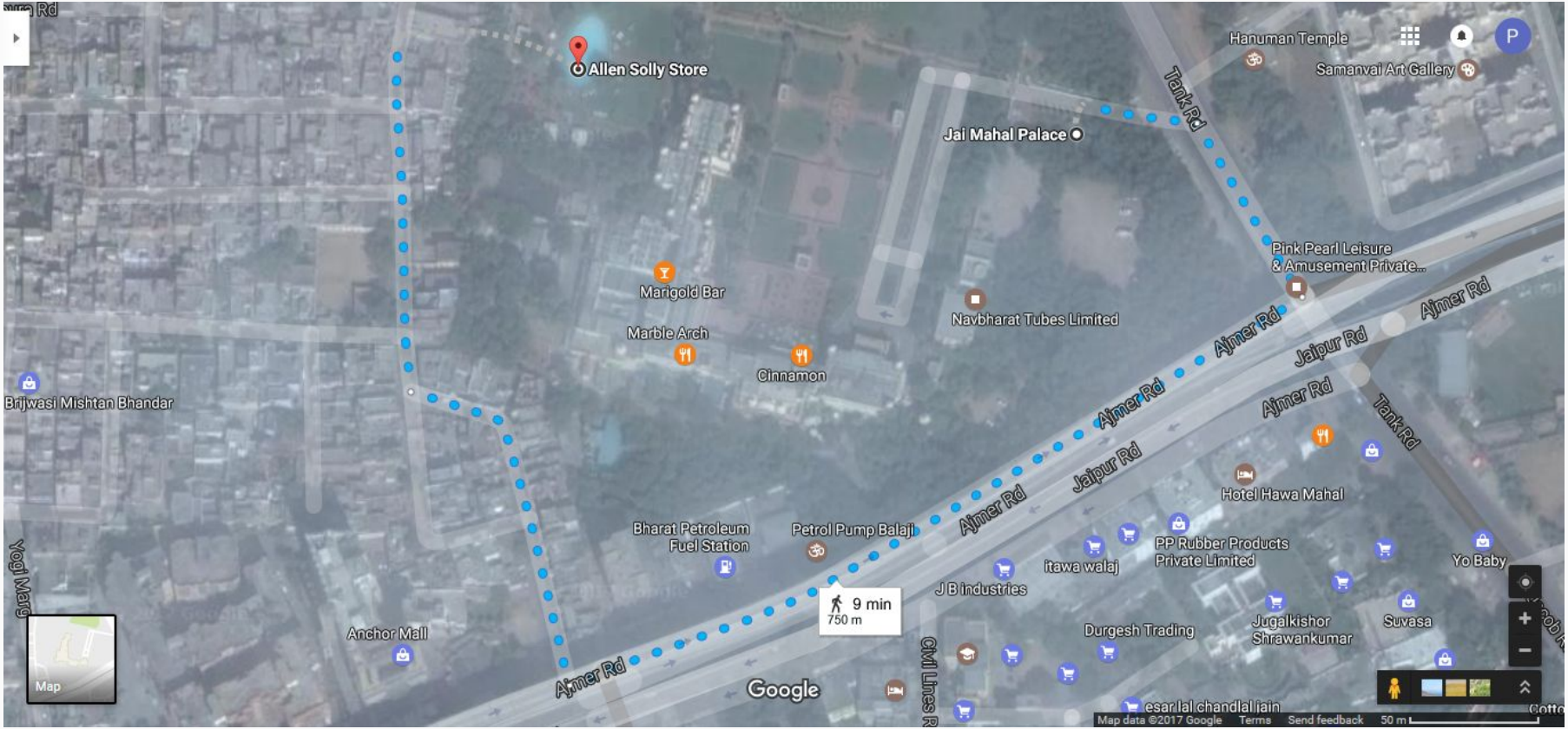
Problem Statement

Correcting Geospatial data using the VGI (Volunteered geographic information) data

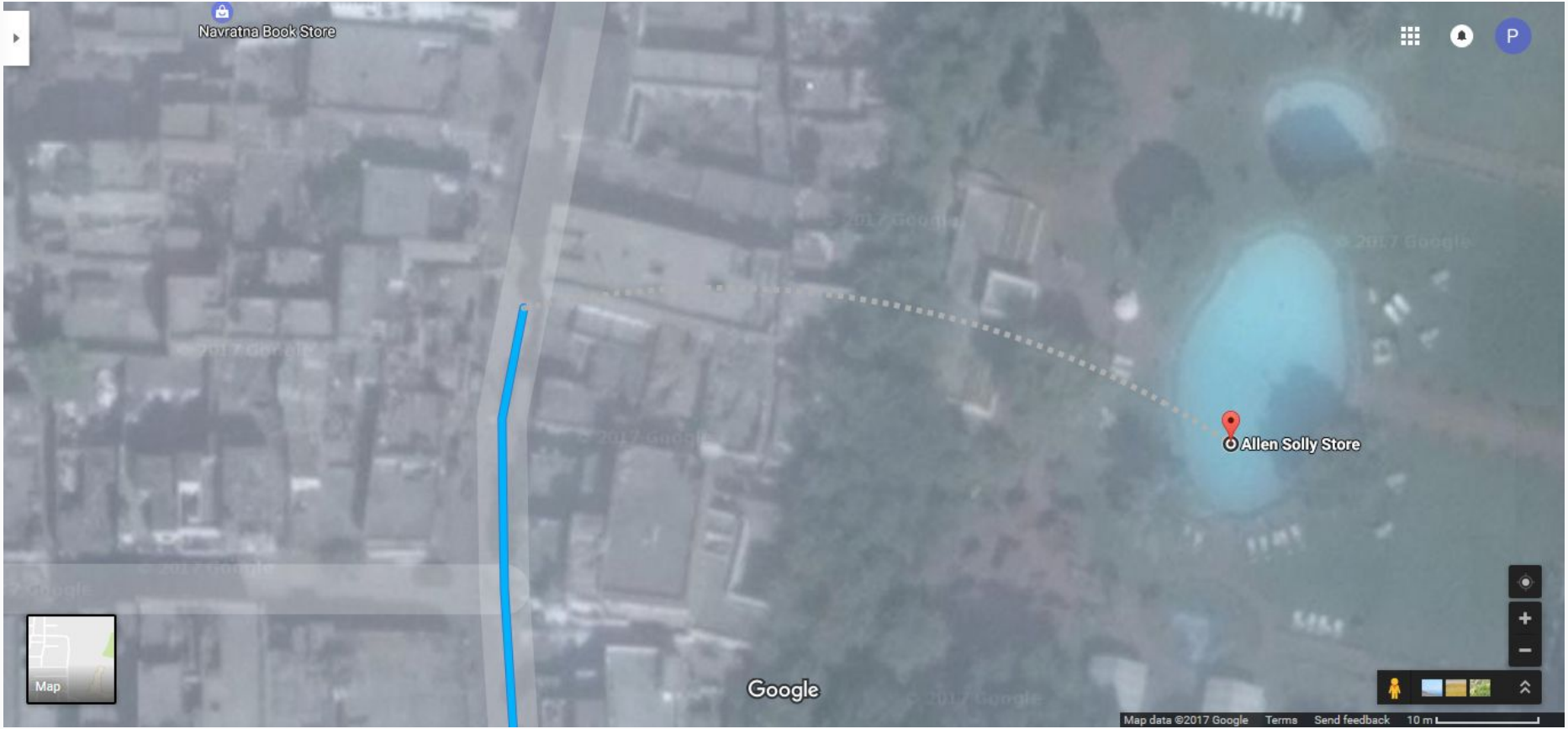
Real life Example of Problem



A clothing store in a swimming pool, *Allen Solly Store Google Maps accessed on 19 Mar 2017*



Let’s see how we can reach this store

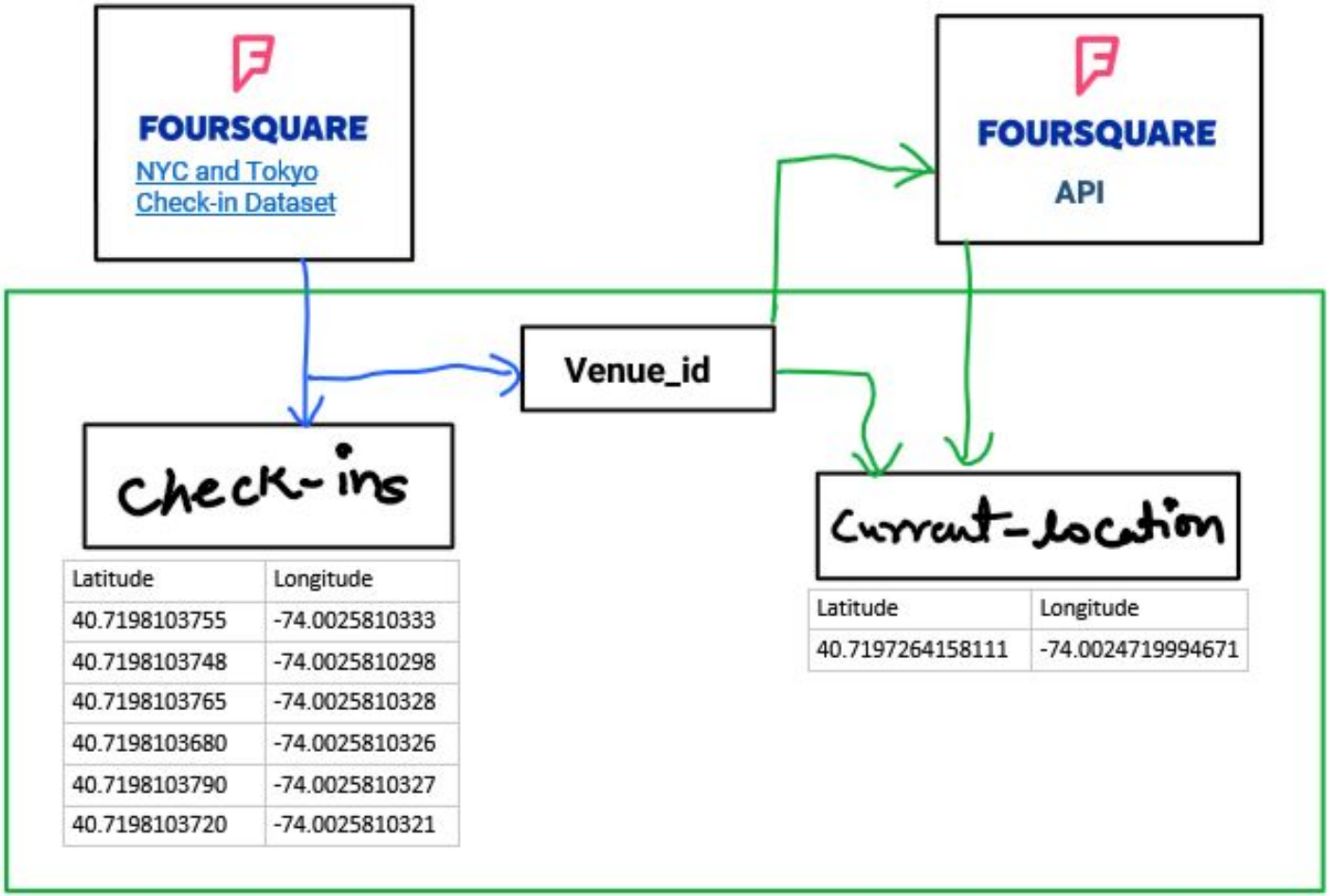


So we have to go through these buildings to reach our desired location

Why This Happened?

Due to incorrect location data of that place

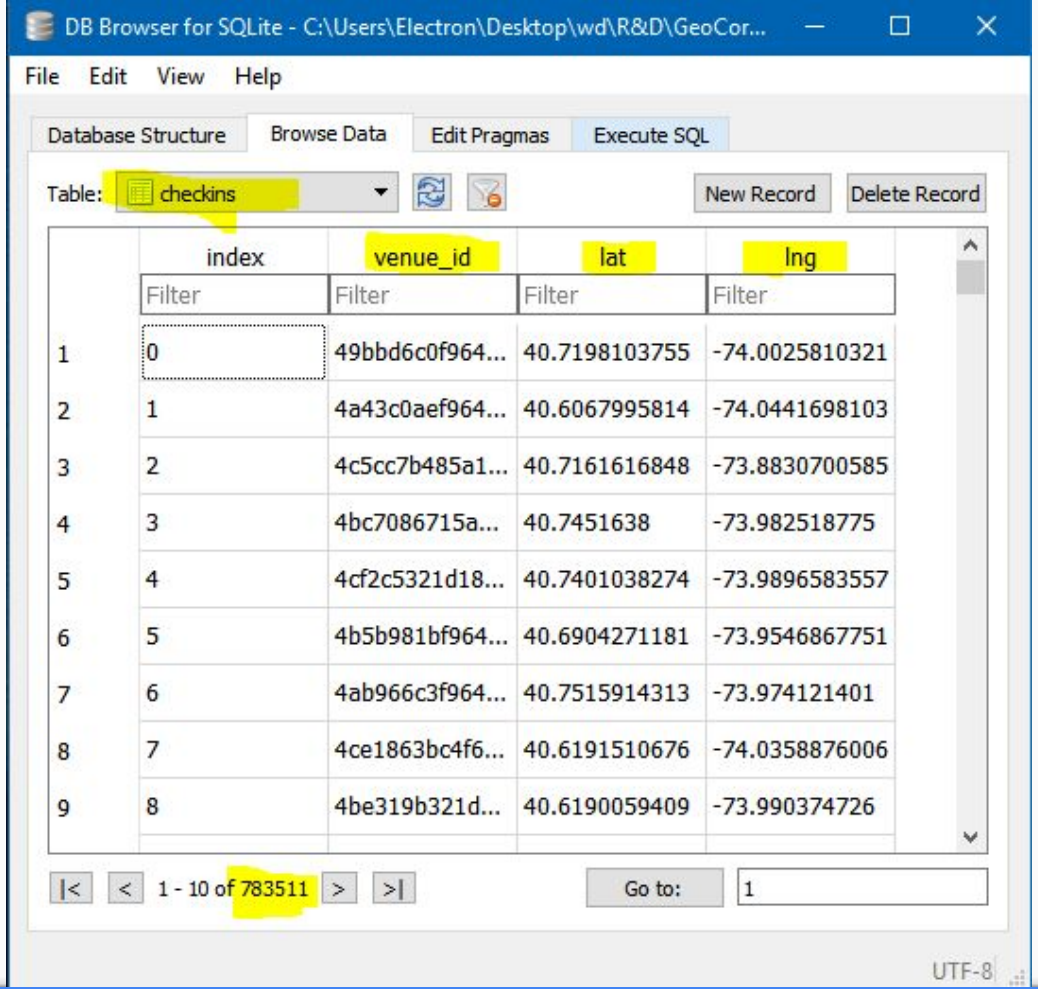
Dataset



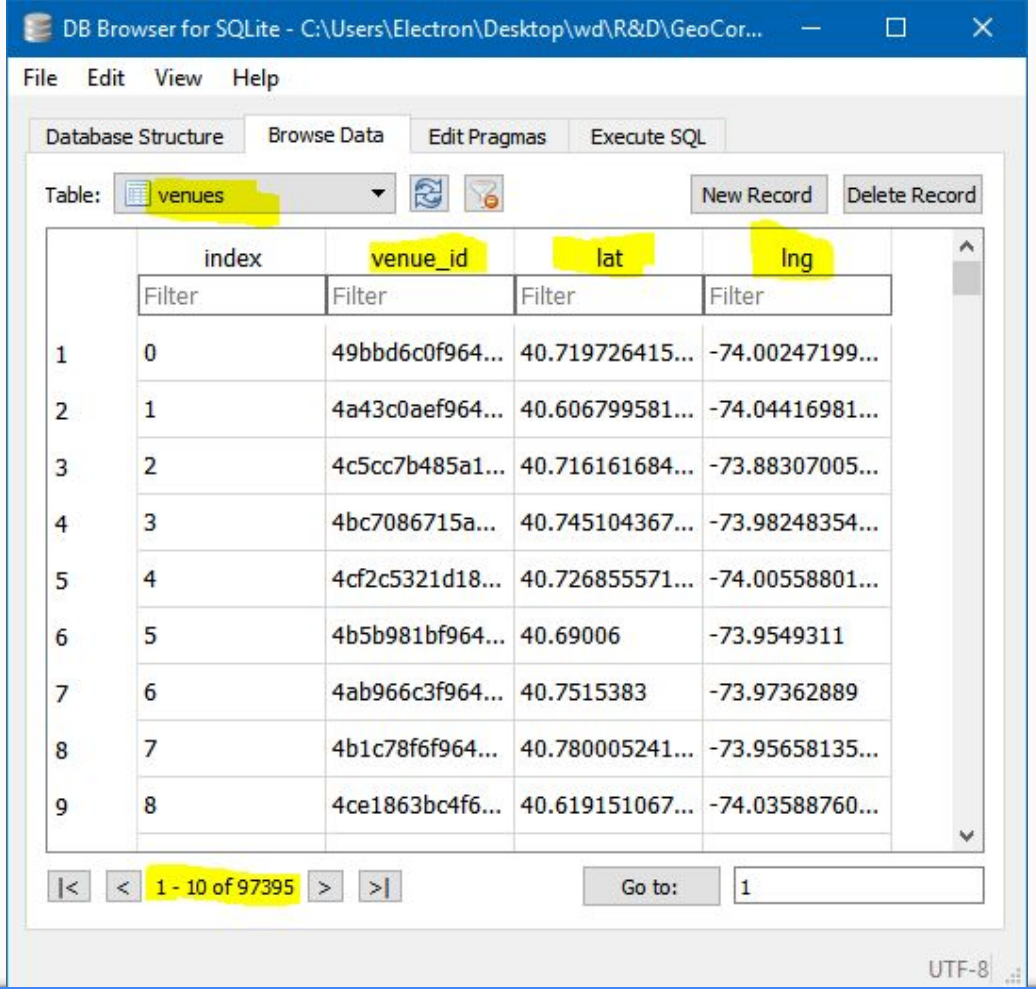
Latitude	Longitude
40.7198103755	-74.0025810333
40.7198103748	-74.0025810298
40.7198103765	-74.0025810328
40.7198103680	-74.0025810326
40.7198103790	-74.0025810327
40.7198103720	-74.0025810321

Latitude	Longitude
40.7197264158111	-74.0024719994671

Fig: creation of dataset using NYC and Tokyo Check-in dataset and Foursquare API



Checkins table : 783511 Checkins

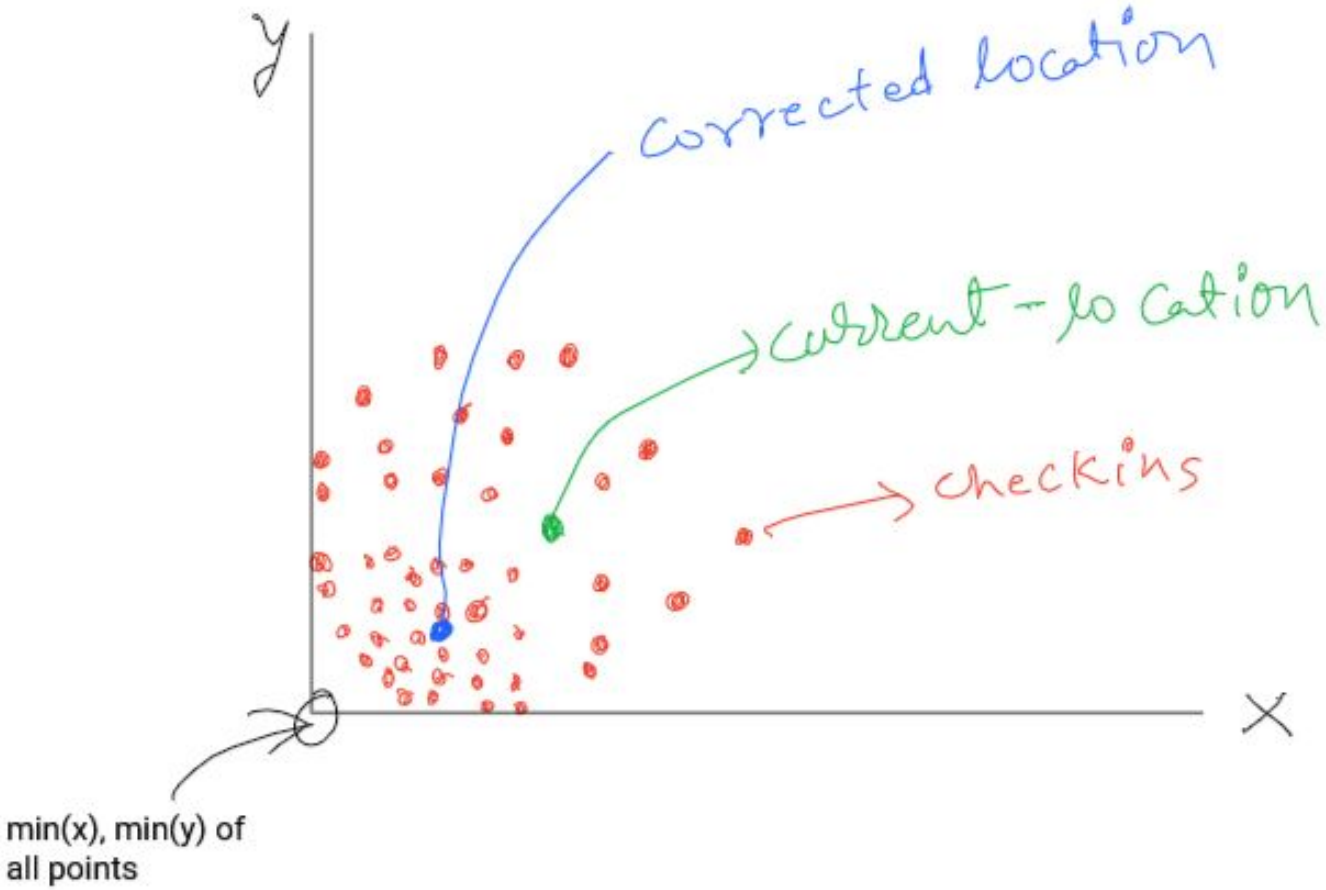


Venue table : 97395 Unique Venues with current location

Algorithm

Algorithm

1. Project 3-D space with check-ins to 2-D using great circle distance
2. Project points on X and Y .
3. Find mode of X and Y, this will give us the new point.
4. Find that point and corresponding coordinates in dataset and find the coordinates latitude and longitude of those X and Y points
5. These coordinates of X and Y will be new corrected location
6. Find the great circle distance between two points.


y
corrected location
current-location
checkins
X
min(x), min(y) of
all points

Calculation of mode

Three cases for calculation of mode

- ID doesn’t exist in check-ins table.
  - Present location is final.
- ID exists and mode can be calculated.
  - Calculated location using mode is final.
- ID exists but multiple mode exist for it.
  - Location closer to current one is treated as final.

Screenshots

```
def _counts(data):
    # Generate a table of sorted (value, frequency) pairs.
    table = Counter(iter(data)).most_common()
    if not table:
        return table
    # Extract the values with the highest frequency.
    maxfreq = table[0][1]
    for i in xrange(1, len(table)):
        if table[i][1] != maxfreq:
            table = table[:i]
            break
    return table

def mode(data):

    table = _counts(data)
    if len(table) == 1:
        return table[0][0]
    elif len(table) > 1 :
        return -1
    elif len(table) == 0:
        return None

def f_mode(data):

    table = _counts(data)
    return table
```

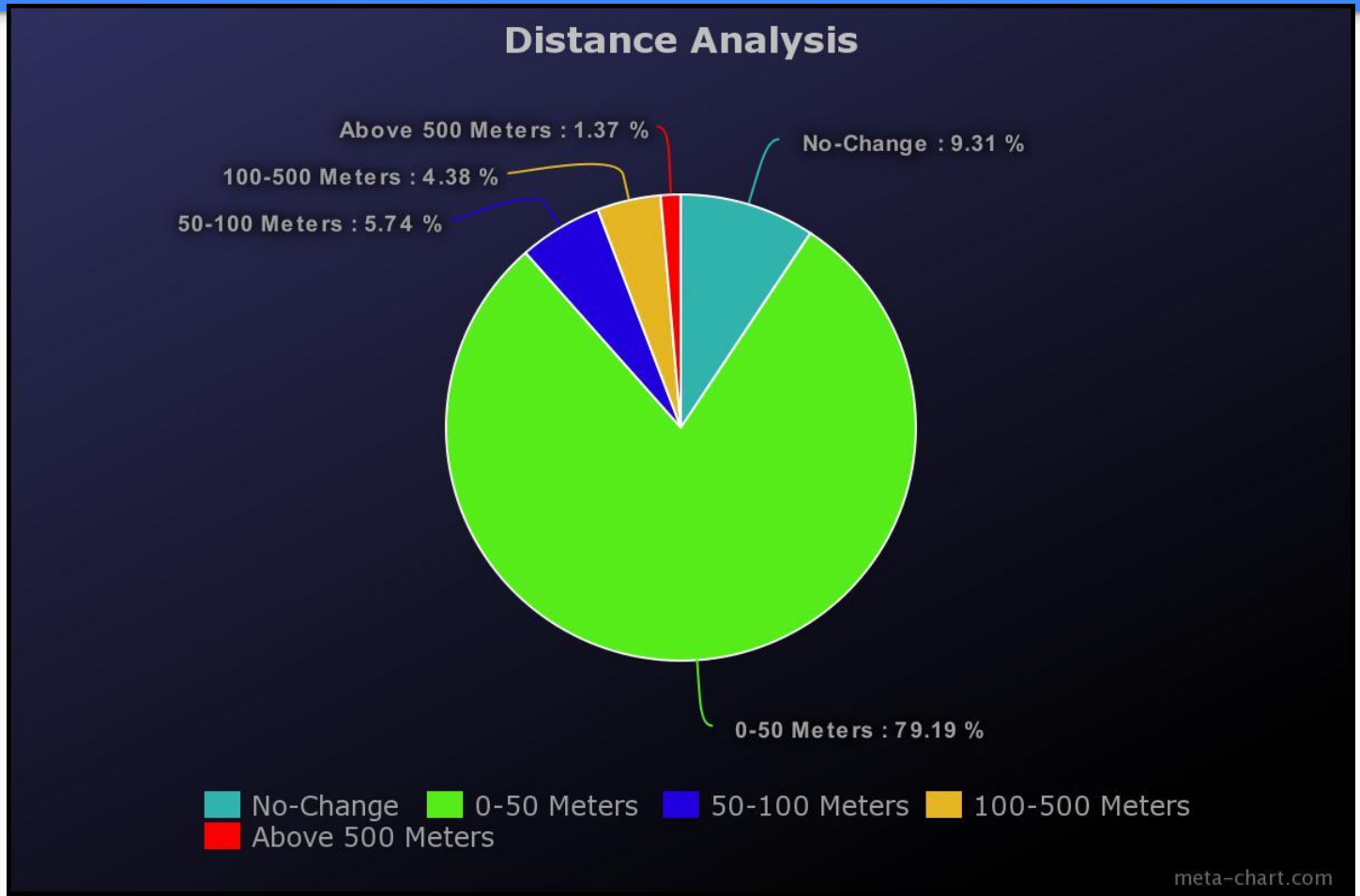
```
        modelon=mode(arlon)
        if modelat is None:
                modelat=vlat

        if modelon is None:
                modelon=vlon

        if modelat is -1:
                list=f_mode(arlat)
                while len(list)>i:
                    if math.pow((float(list[i][0])-float(vlat)),2) < min :
                        min=list[i][0]
                    i+=1

                while len(list) > 0 :
                    list.pop()

                modelat=min
        i=0
        min=999
        if modelon is -1:
                list=f_mode(arlon)
                while len(list)>i :
                    if(math.pow(float(list[i][0])-float(vlon),2) < min):
                        min=list[i][0]
                    i+=1

                while len(list) > 0 :
                    list.pop()
```

Results

Analysis of Results

- Total number of locations (id’s) in database = 97395
- Total number of locations with no change = 9070
- Total number of locations with a change = 88325
- Roughly 90% locations were found out to be inaccurate.

Distance Distribution



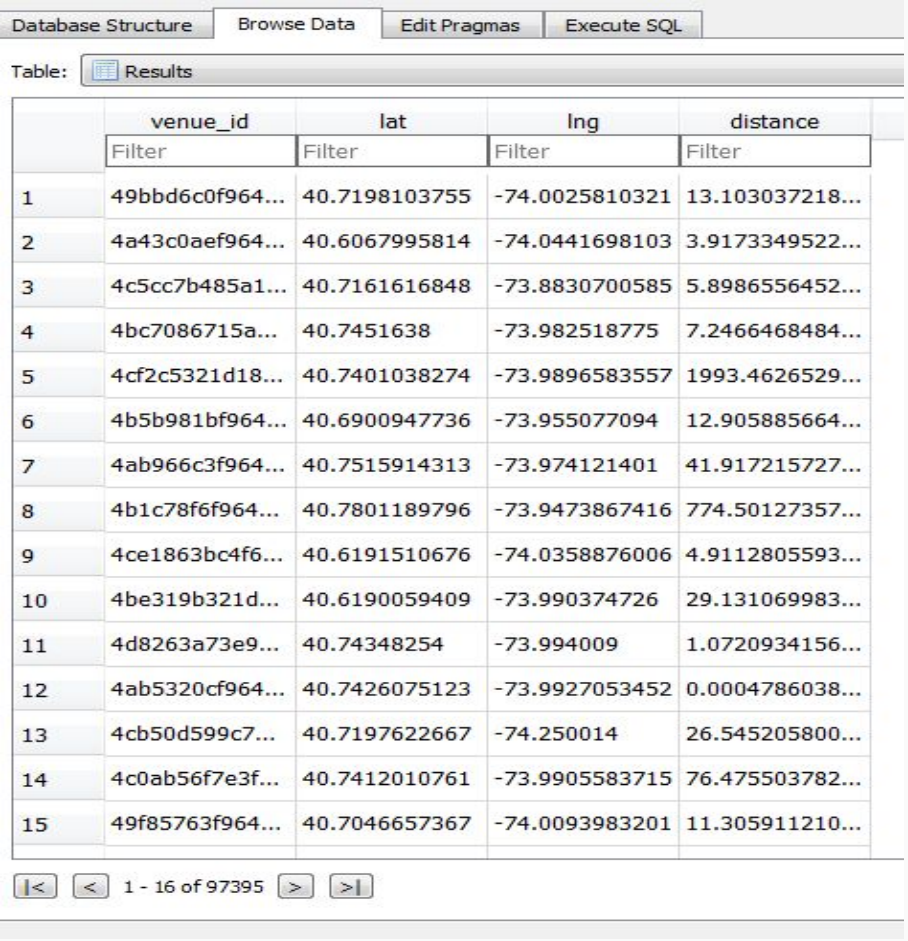
In depth

- 0-50 meters = 77131
- 50-100 meters = 5591
- 100-500 meters = 4268
- Above 500 meters = 1335
- Average change in distance =258 meters

Outcome of Analysis

- Even If we don’t consider the change in distance of less than 100 meters, still around 6% of location need to be changed which is quite a significant proportion.

- And when we want to build maps for self-driving cars the locations with less than 100 meters change need to be considered as well.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Results

	venue_id	lat	lng	distance
	Filter	Filter	Filter	Filter
1	49bbd6c0f964...	40.7198103755	-74.0025810321	13.103037218...
2	4a43c0aef964...	40.6067995814	-74.0441698103	3.9173349522...
3	4c5cc7b485a1...	40.7161616848	-73.8830700585	5.8986556452...
4	4bc7086715a...	40.7451638	-73.982518775	7.2466468484...
5	4cf2c5321d18...	40.7401038274	-73.9896583557	1993.4626529...
6	4b5b981bf964...	40.6900947736	-73.955077094	12.905885664...
7	4ab966c3f964...	40.7515914313	-73.974121401	41.917215727...
8	4b1c78f6f964...	40.7801189796	-73.9473867416	774.50127357...
9	4ce1863bc4f6...	40.6191510676	-74.0358876006	4.9112805593...
10	4be319b321d...	40.6190059409	-73.990374726	29.131069983...
11	4d8263a73e9...	40.74348254	-73.994009	1.0720934156...
12	4ab5320cf964...	40.7426075123	-73.9927053452	0.0004786038...
13	4cb50d599c7...	40.7197622667	-74.250014	26.545205800...
14	4c0ab56f7e3f...	40.7412010761	-73.9905583715	76.475503782...
15	49f85763f964...	40.7046657367	-74.0093983201	11.305911210...

|< < 1 - 16 of 97395 > >|

Table: Results

	venue_id	lat	lng	distance
	Filter	Filter	Filter	Filter
1269	4e3009bf1838...	40.7636621371	-74.0233879188	22.708571251...
1270	4cc8a5dfe792...	40.7612853775	-73.9926817602	134.07469063...
1271	4e54e3a17d8...	40.6652116607	-73.9418603428	5.0389340784...
1272	4c2f7a5666e4...	41.112794142...	-74.16263401...	0.0
1273	4a8d57e7f964...	40.7070756342	-74.010643959	164.72906666...
1274	4c360f57dfb0...	40.8389153916	-74.1807540652	45.826481531...
1275	4e613585813...	40.8278390826	-73.925768137	3.4763202670...
1276	4dffc29d1495...	40.763175403	-74.0237035049	6.0457569701...
1277	4a3ffdc5f964a...	40.7370985738	-73.9903976817	101.64766170...
1278	4b7ce0f6f964...	40.8703651645	-74.1989564896	5.7961305798...
1279	4af18817f964...	40.7360635917	-74.0293872356	30.275360094...
1280	4b720649f964...	40.8707789293	-74.198141098	2.9300624819...
1281	4af3318cf964...	40.5952441793	-73.9551571012	6.4506246409...
1282	4b0dc4abf964...	40.878445	-73.921659	1.0699149560...
1283	4de5ae6dd4c...	40.6759551617	-73.9237342759	4.7323041173...

Results table : 97395 corrected locations

Observations

- Algorithm works best when we have massive amount of check-in data
- Algorithm is tolerant to outliers and simple to implement in comparison with Algorithm we designed at first stage
- Algorithm will never introduce more error then current error, provided that check-in data is correct

Visualisation Of Results

Technical Stack of Visualisation Toolkit

Technical Stack of Visualisation Toolkit

Visualisation View

GeoCorrect

Correction of maps(location data) using on VGI data

Visit https://github.com/electron0zero/GeoCorrect for more info

Click on Map Icon to see data on map

Click on Table Icon to see data in tabular form

Table	Map	Foursquare Venue ID	Old Latitude	Old Longitude	New Latitude	New Longitude
		49bbd6c0f964a520f4531fe3	40.719726415811074	-74.00247199946715	40.7198103755	-74.0025810321
		4a43c0aef964a520c6a61fe3	40.606799581406435	-74.04416981025437	40.6067995814	-74.04416981029999
		4c5cc7b485a1e21e00d35711	40.71616168484322	-73.88307005845945	40.7161616848	-73.88307005850001
		4bc7086715a7ef3bef9878da	40.74510436721173	-73.98248354107581	40.7451638	-73.982518775
		4cf2c5321d18a143951b5cec	40.726855571680865	-74.00558801400682	40.7401038274	-73.9896583557
		4b5b981bf964a520900929e3	40.69006	-73.9549311	40.6900947736	-73.95507709399999
		4ab966c3f964a5203c7f20e3	40.7515383	-73.97362889	40.7515914313	-73.974121401
		4b1c78f6f964a520b40724e3	40.78000524198429	-73.95658135414124	40.7801189796	-73.9473867416
		4ce1863bc4f6a35d8bd2db6c	40.61915106755737	-74.03588760058483	40.6191510676	-74.0358876006
		4be319b321d5a59352311811	40.618838900608	-73.99010896682735	40.6190059409	-73.990374726

Home Map

Venue Information

Foursquare Venue ID	49bbd6c0f964a520f4531fe3
Name	Pearl Art & Craft Supply
Address	308 Canal St (btwn Broadway & Mercer), New York, NY 10013, United States
Foursquare URL	https://foursquare.com/v/pearl-art--craft-supply/49bbd6c0f964a520f4531fe3

New Location

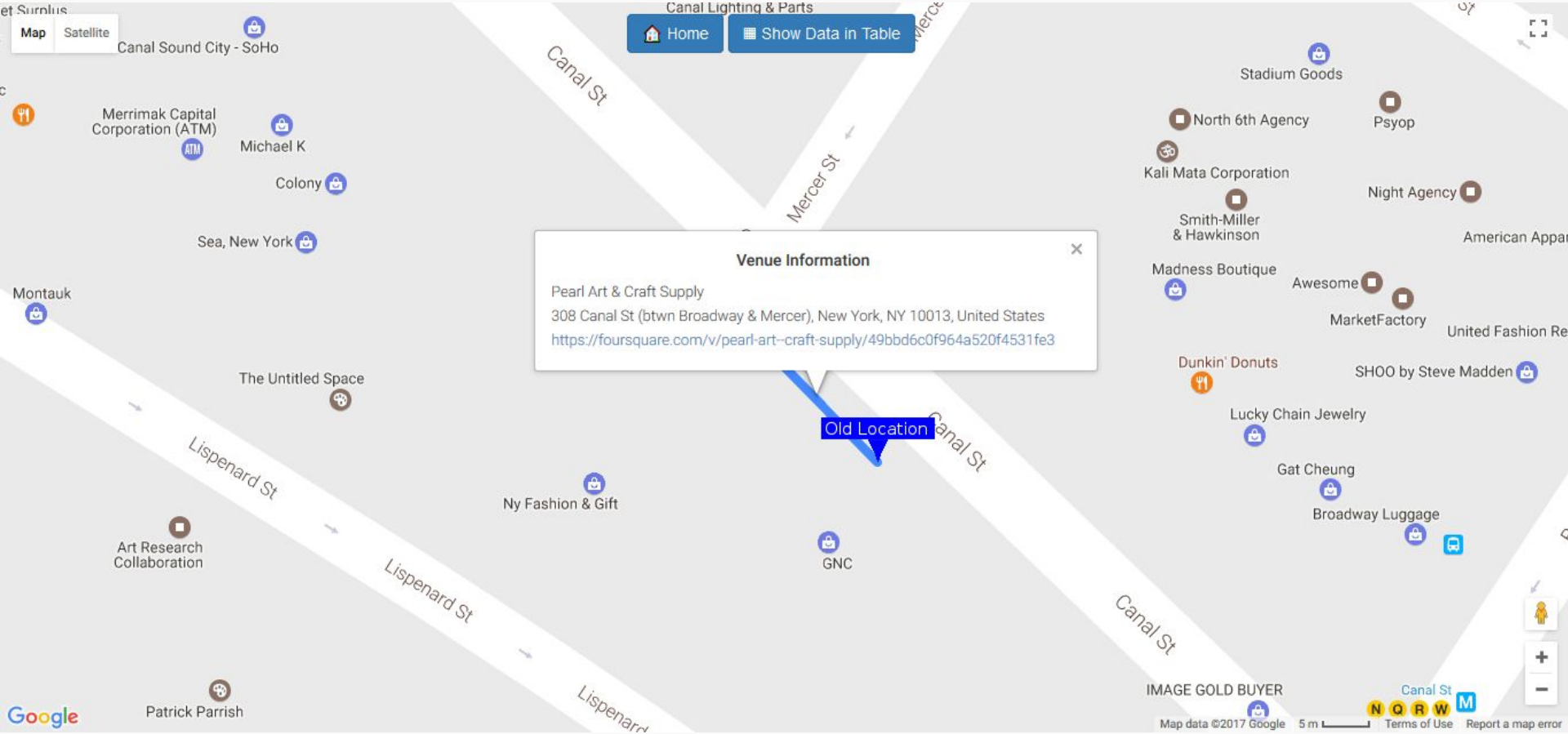
Latitude	Longitude
40.7198103755	-74.0025810321

Old Location

Latitude	Longitude
40.719726415811074	-74.00247199946715

Checkins

Latitude	Longitude
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321


Map
Satellite
Home
Show Data in Table
Canal Lighting & Parts
Canal Sound City - SoHo
Merrimak Capital Corporation (ATM)
Michael K
Colony
Sea, New York
Montauk
The Untitled Space
Art Research Collaboration
Patrick Parrish
Ny Fashion & Gift
GNC
Canal St
Mercer St
Lispenard St
Stadium Goods
North 6th Agency
Psyop
Kali Mata Corporation
Night Agency
Smith-Miller & Hawkinson
Madness Boutique
Awesome
MarketFactory
Dunkin' Donuts
SHOO by Steve Madden
Lucky Chain Jewelry
Gat Cheung
Broadway Luggage
IMAGE GOLD BUYER
Venue Information
Pearl Art & Craft Supply
308 Canal St (btwn Broadway & Mercer), New York, NY 10013, United States
https://foursquare.com/v/pearl-art--craft-supply/49bbd6c0f964a520f4531fe3
Old Location
Google
Map data ©2017 Google
5 m
Terms of Use
Report a map error


304 Canal St
New York
View on Google Maps
Home
Show Data in Table
212-300-3504
212-431-7932
Old Location
New Location
Google
© 2017 Google
Terms of Use
Report a problem

Demo

http://geocorrect.herokuapp.com/

Future of this project

Future of this project

- On 4th April 2017,Google shut down its current procedure for map correction.
- We have an idea about how our algorithm can be used to solve this problem in a efficient and sustainable way.
- The only requirement for our algorithm to work is a correct dataset and google has all the resources required to come up with one.

Idea

- So , how can Google create a dataset with correct locations of all erroneously marked location.
- Google can launch an app in which any individual can apply for a change in location for any given location.
- Google keeps track of location of G-Maps users.So,whenever there is a request for a location change Google can send out survey form to 15-20 random people who live near that location for authentication of the query.

Idea

- If all people agree to the change in location then google should proceed and update the dataset for a change.
- Once the locations to be updated are in the dataset, our algorithm can be used to update the locations.
- And this approach can be used to attain a dynamic map correction and this can make Google map as accurate as possible.

Why People will file query ?

- Google can provide people who post a correct location change with incentives like space in Google Drive or benefit on google products.
- They can provide them with some artificial money say credit points which can be used to get benefit from any of google services.
- We think If google uses this approach Google Maps will never ever have any competitor.

Difficulties

Difficulties

- Had to throw away all the work done in first few weeks due to anonymized foursquare dataset
- Not able find a non-anonymized dataset with all the attributes required
- Had to build a dataset using the attributes from NYC and Tokyo Check-in Dataset and Used the Foursquare API to fetch other Required attributes

Difficulties

- Algorithm we designed at first was prone to outliers and hence not so accurate, that resulted in redesign of algorithm
- During Deployment of Visualisation ToolKit, we have to use Mobile data because SSH, Git & FTP ports are blocked on University Network

References

Xavier, Emerson, Francisco J. Ariza-López, and Manuel A. Ureña-Cámara. "A Survey of Measures and Methods for Matching Geospatial Vector Datasets." *ACM Computing Surveys (CSUR)* 49.2 (2016): 39.

Bruns, Tom, and Max Egenhofer. "Similarity of spatial scenes." Seventh international symposium on spatial data handling. Delft, The Netherlands, 1996.

Kolahdouzan, M. R., et al. "GeoMatchMaker: automatic and efficient matching of vector data with spatial attributes in unknown geometry systems." *Proc., UCGIS 2005 Summer Assembly* (2005).

References

Du, Heshan, et al. "Geospatial information integration for authoritative and crowd sourced road vector data." *Transactions in GIS* 16.4 (2012): 455-476.

Geocrowdsourcing and accessibility for dynamic environments H Qin, RM Rice, S Fuhrmann, MT Rice, KM Curtin… - GeoJournal, 2016

Zook, Matthew, et al. "Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake." World Medical & Health Policy 2.2 (2010): 7-33.

References

"API Endpoints". *Developer.foursquare.com*. N.p., 2017. Web. 20 Mar. 2017.

Dingqi Yang, et al. "Modeling User Activity Preference By Leveraging User Spatial Temporal Characteristics In Lbsns". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.1 (2015): 129-142. Web.

"Earth Ellipsoid". *En.wikipedia.org*. N.p., 2017. Web. 20 Mar. 2017.

"Earth Radius". *En.wikipedia.org*. N.p., 2017. Web. 20 Mar. 2017.

"Electron0zero/Geocorrect". *GitHub*. N.p., 2017. Web. 20 Mar. 2017.

References

"Geopy/Geopy". *GitHub*. N.p., 2017. Web. 20 Mar. 2017.

Illustration Of Great-Circle Distance.. 2017. Web. 20 Mar. 2017.

“Calculating Latitude/Longitude X Miles From Point?". *Gis.stackexchange.com*. N.p., 2017. Web. 20 Mar. 2017.

"Understanding Latitude And Longitude". *Learner.org*. N.p., 2017. Web. 20 Mar. 2017.

"Google Maps Embed API | Google Developers". Google Developers. N.p., 2017. Web. 26 Apr. 2017.

References

"Welcome | Flask (A Python Microframework)". Flask.pocoo.org. N.p., 2017. Web. 26 Apr. 2017.

"Mbr/Flask-Bootstrap". GitHub. N.p., 2017. Web. 26 Apr. 2017.

"Flask-Googlemaps 0.2.4 : Python Package Index". Pypi.python.org. N.p., 2017. Web. 26 Apr. 2017.

Questions?

Thanks