Open Street Map Project Report

Map Area

Manhatten, NY, United States

- https://mapzen.com/data/metro-extracts/your-extracts/54783bc61715 (link to download)

I was always fascinated by this posh area and this was the chance to get insights to that area. This was not initially present in the mapzen site but I was provided the customised data on request.

Problems Encountered in the Data

After downloading the data in OSM format which was around 100 MB which I downsized as a sample from the code given in project overview keeping K factor at 100 . The sample data was pretty much clean so I had to apply all the code on the full file. The problems spotted in data were:

- The most common was inconsistency and abbreviation of street names For example Street was represented in multiple abbreviations like st , St. and typo errors like street.

- The second problem was on post codes with extensions like NY and NJ attached as well as some phone number added with that key .

- Also the postal codes , some of them were not of manhatten starting with 07 as for Manhatten it starts from 10 in series and further. Eg. NY 10012, (718) 778-0140, NJ 07024

- While catching tags with problematic characters , I printed out the problem tags which belonged almost to the city racks cycle service with a '.' In the tag. But I have kept it out of the database anyway because I feel it won't give any significant insight.

Functions involved in solving the street type problem

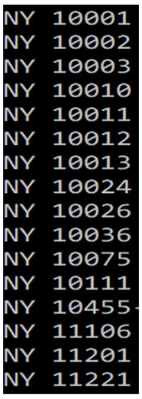
```
def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

update_name( mapping,name):

        m = street_type_re.search(name)

        street_type = m.group()
        if street_type  in mapping :
            better_name = re.sub(street_type_re, mapping[street_type], \
                                    name)
            print  name, "=>", better_name
            return better_name
        else :
               return name
```

The first function is used to identify the correct tag type for the street name out of all and the second function “update name “ is used to replace it from the look up mapping list declared in the global scope.

Functions involved in cleaning post codes

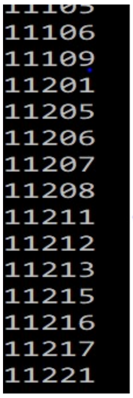
```
def is_post_code(elem):
    return (elem.attrib['k'] == "addr:postcode")

def correct_post_code(elem):

        m = code_type_re.match(elem.attrib['v'])
        if m:
            print elem.attrib['v']
            if elem.attrib['v'] [0] == "(" :
                correct_code = elem.attrib['v'][6:]
                print correct_code
                return correct_code
            else:
                correct_code = re.sub(code_type_re,"" , elem.attrib['v'])
                print correct_code
                return correct_code
        else :
                return   elem.attrib['v']
```

First function “is_post_code” catches the tag with key value of “post code” and the second function finds the problematic characters with re function in the string and corrects it to a better format.



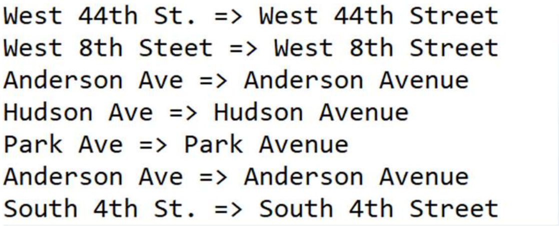
```
NY 10001
NY 10002
NY 10003
NY 10010
NY 10011
NY 10012
NY 10013
NY 10024
NY 10026
NY 10036
NY 10075
NY 10111
NY 10455
NY 11106
NY 11201
NY 11221
```

Figure 1 before cleaning



```
11106
11109
11201
11205
11206
11207
11208
11211
11212
11213
11215
11216
11217
11221
```

Figure 2 after cleaning



```
West 44th St. => West 44th Street
West 8th Steet => West 8th Street
Anderson Ave => Anderson Avenue
Hudson Ave => Hudson Avenue
Park Ave => Park Avenue
Anderson Ave => Anderson Avenue
South 4th St. => South 4th Street
```

Figure 3 console output while auditing street type

Over view of the size of files

```
manhatten.osm ......... 462 MB
manhatten.db .......... 358 MB
nodes.csv ............. 163 MB
nodes_tags.csv ........ 11.3 MB
ways.csv .............. 20.2 MB
ways_tags.csv ......... 62.2 MB
ways_nodes.csv ......... 52.2 MB
```

Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
```

```
1870329
```

Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
```

```
317870
```

Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;

2562
```

Top 10 contributing users

```
sqlite> SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 10;

Rub21_nycbuildings             1076245
Minewman                       132699
Ediyes_nycbuildings            119900
Smlevine                       103988
Ingalls_nycbuildings           101770
Lxbarth_nycbuildings           89875
Robgeb                         68063
celosia_nycbuildings           54004
korzun                         34145
mikercpc                       20374
```

Number of users appearing only once

```
867
```

Contributor statistics

- Top contributor makes 49.2 per cent of the data
- Top 10 users contribute to 82.3 per cent of the data
- 99.6 per cent of users (2552) contribute to only remaining 17.7 per cent of the data

I read about the gamification strategy in the sample document but what era we are living in, google maps which is a free service has created such a fine organisation of street map data which can now be updated by users in a simpler UI format of yes or no questions
So as per my views the development of this project as a viable commercial amenity might never be possible.

Other Explorations

Most number of tag key values of a given type in node tags :

```
sqlite> select count(key), key from nodes_tags group by key order by
count(*) desc limit 100;

62070|housenumber
62065|street
60574|postcode
13464|name
12962|amenity
9788|highway
5233|created_by
4419|capacity
4350|city
```

Some fun insights found a manholes and atm keys in the tags

No. of places having atm service :

```
sqlite> select value , count(*) from nodes_tags where key = 'atm' and value
='yes';

yes|169
```

No. of manholes in the area :

```
sqlite> select value,count(*) from nodes_tags where key = 'manhole' group
by value;
drain|1
unknown|50
water|1
yes|51
```

the tag has been used only once in a node as I cross checked the value by joining it with distinct node ids.

Most popular sports activites:

```
sqlite> select value , count(*) from nodes_tags where key = 'sport' group
by value order by count(*) desc limit 15;
chess|7
fitness|7
basketball|6
cycling|5
```

```
yoga|5
athletics|4
table_tennis|4
tennis|4
baseball|3
running|3
skating|3
swimming|3
climbing|2
crossfit|2
dance|2
```

this data doesn’t seem to be realistic because of the count value of the fields is weak to establish confidence on it and also the fact that football which is a popular sport is nowhere to be found.

Most amenities in the area :

```
sqlite> select value , count(*) from nodes_tags where key = 'amenity' group
by value order by count(*) desc limit 15;

bicycle_parking|4243
restaurant|2069
cafe|779
school|686
fast_food|555
place_of_worship|479
bicycle_rental|409
bar|386
bank|333
bench|327
drinking_water|269
post_box|217
pharmacy|206
embassy|160
fire_station|156
```

on seeing the whole list there were some important points like number of hospitals and pharmacies

Finding average speed in the area :

```
sqlite> select value, count(*) from nodes_tags where key = 'maxspeed' group
by value order by count(*) desc ;

25 mph|2
35 mph|1
40 mph|1
45 mph|1
```

Taking average the avg speed comes out to be 34 mph but we cant make it a good assumption as the number of samples is less.

Top fast_food chain with most branches：

```
Sqlite> SELECT  nodes_tags.value , count(*) FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='fast_food') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'  GROUP BY nodes_tags.value ORDER BY count(*)
desc LIMIT 10;
McDonald's|52
Subway|47
Dunkin' Donuts|19
Chipotle|16
Burger King|9
Chipotle Mexican Grill|9
Shake Shack|9
Five Guys|6
Pret A Manger|6
Taco Bell|6
```

This gives you the idea that most eaten fast food will be subway and mc Donalds

No. of cities in the data :

```
sqlite> select value , count(*) from nodes_tags where key ='city' group by
value order by count(*) desc;

New York|2667
Brooklyn|733
Hoboken|502
New York City|140
Astoria|84
Bronx|42
Long Island City|31
Jersey City|30
Union City|12
Queens|10
```

The list is very long including many other areas as it was pointed out by post codes that lot of territories which are not in manhatten are also in the data making it impure. This makes me suggest more calibrated gps tracker to improve the data on the basis of this query.

Finding top 3 religions in the area :

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE
value='place_of_worship') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;
```

```
christian|403
jewish|25
muslim|8

As expected
```

Top 10 most popular cuisines in the manhatten area :

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC LIMIT 10;

italian|151
american|97
pizza|97
mexican|86
chinese|72
japanese|58
french|48
indian|48
thai|48
burger|41
```

clearly Italian seems to be favourite of the americans

Conclusion

After the completion of this whole project on manhatten I feel that data is relatively cleaner compared to other parts of the world. But again it has some drawbacks specially with impurities like Brooklyn and Hoboken data is inside this file to much extent which disrupts the uniformity .
Also there is a lack of information like if I wanted to calculate the average of maximum speed of vehicle there were not enough entries to get a firm idea.
But again a person can understand a lot and get insights to what the city is about. If I want to open a food business would I open a fast food joint or a proper dine in restaurant ??
Well the trends in the data suggest that there are lot more fast food joints than the restaurants. But again this is only one aspect of the decision making.
Improvement Solution :
what ideas I would suggest for improvements is to use better calibrated GPS devices to do the survey or maybe use cell phone data which can automatically

input the co ordinates , time stamp, node id and all the survey people have to do is to attach tags with information about the places.
Also we can provide a UI based website or app with yes\no questions to add tags to the node. This type of arrangement can help common people to contribute to the data easily rather coding it in xml format which will be done by the back end of the project.

Benefits:

- Better and calibrated GPS systems may provide a cleaner and more accurate data.
- Less manual labour involoved in collecting the data nad less labour eventually helps in cost cutting
- The UI based idea will help generate a larger and detailed database autonomously

Anticipated issues :

- Better tracking equipment increase the budget
- Sometimes the trackers which are currently used might not be at fault and there might be a fault with connectivity between the link to the sat from GPS which might not get solved with using better and more expensive tracker
- For the UI based idea a team of coders is to be invested upon increasing the budget of the project.