OpenStreetMap Data Case Study

Map Area

San Jose, CA

- https://www.openstreetmap.org/relation/112143 (https://www.openstreetmap.org/relation/112143)

I chose this area because this is the city that I grew up in. I already have preconceived notions about this city so I am interested in seeing if database queries will reveal new information. Also I hope auditing the map data will allow me to contribute fixes to OpenStreetMap.

Problems Encountered in the Map Data

Originally I audited the OSM file for street names only. Upon the first pass through and building the database and then performing a few preliminary queries I found other areas to be audited. I will be addressing the following issues:

- Inconsistent street name abbreviations ("Los Gatos Blvd.", "Los Gatos Boulevard")
- Street names not ending with common names ("Saratoga", "Blossom Hill")
- Incosistent postal codes ("95014", "CA 95110", "95014-2522")
- Problematic postal codes (Some tags had a postal code value of a city name rather than the a 5 digit post code, i.e. "CUPERTINO")

Inconsistent Street Name Abbreviations

The initial auditing of the map data revealed that there are inconsistencies in street name abbreviations. For the most part the street names are not abbreviated so words like "Street" and "Boulevard" are fully written out. However there are a handful of instances where these words are abbreviated and abbreviated in different ways. For example, "Street" can be abbreviated as "St" or "St.". I opted to fix this issue using regular expressions exactly like how it was shown in the class exercises. This of course required me to do the initial audit and find all of the various errors of this type and create a mapping.

def update_name(name, mapping):

```
m = street_type_re.search(name)
if m:
    street_type = m.group()
    if street_type not in expected:
        name = re.sub(street_type_re, mapping[street_type], name)

return name
```

mapping = { "St": "Street", "St.": "Street", "street":"Street", "Rd": "Road", "Rd.": "Road", "Ave": "Avenue", "ave": "Avenue", "Hwy": "Highway", "court": "Court", "Sq": "Square", "Blvd": "Boulevard", "Boulvevard": "Boulevard", "Blvd.": "Boulevard", "Ln": "Lane", "Dr": "Drive", "Cir": "Circle", "Ct" : "Court", "Pkwy": "Parkway"}

Street Names Not Ending with Common Names

From my initial audit of the street names I also found that San Jose has a lot of instances where the street name does not end with a common word such as "Street", "Boulevard", "Avenue" etc. This was a very simple fix as these street names (i.e. Blossom Hill) are proper and should be excused in the code. This just required a proper list of expected street names

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Loop", "Circle", "Square", "Lane", "Road", "Trail", "Parkway", "Commons", "Way", "Terrace", "Highway","Expressway", "East", "West", "Bellomy", "Winchester", "Oro", "1","Esquela", "Bascom", "6", "Plaza","Walk","Portofino","Napoli","Paviso","Barcelona","Volante","Sorrento","Franklin","Real", "Julian", "Flores", "Saratoga", "0.1","7.1", "Presada", "Row", "Alley", "Alameda", "Seville", "Montaña", "Palamos", "Marino", "Oaks", "Luna", "Madrid", "Mall", "Hamilton", "81", "114", "Robles", "Hill"]

Inconsistent and Problematic Postal Codes

Originally I compiled a database after only cleaning the street names. After initial sql queries I found that postal codes had a few problems. I audited the original OSM file looking for problematic zip codes. Most zip codes are 5 digits long while other zip codes had the 4 digit extension as well. Some zip codes has the state Abbreviation at the beginning. I cleaned the postal code by just captuing the main 5 digit postal code. The last type of problem I encountered was where the user inputed the postal code value with the county name rather than the 5 digit code. This particular problem only happened in the city, Cupertino, so I made the function replace the string "Cuptertino" with the 5 digit zip 95014.

```
def update_postcode(postcode):

if postcode == 'CUPERTINO':
    clean_postcode = 95014
else:
    search = re.match(r'^\D*(\d{5}).*', postcode)
    clean_postcode = search.group(1)
return clean_postcode
```

Exploration on Possible Problems

From the previous investigation on postal codes shows that there are postal codes for surrounding cities around San Jose. This made me wonder what other cities are included in the map data so I performed a query to find the different cities included in the map data.

```
sqlite> select tags.value, count(*) as count

...> from (select * from nodes_tags
...> UNION ALL
...> select * from ways_tags) tags
...> where tags.key like '%city'
...> group by tags.value
...> order by count DESC;
```

Here are the results (top 9):

```
Sunnyvale|3417
San Jose|914
Morgan Hill|395
Santa Clara|318
Saratoga|233
San Jos├-|164
Milpitas|102
Los Gatos|100
Campbell|76
```

I can see here that all of the cities surrounding San Jose are represented. This prompted me to go back to https://mapzen.com/data/metro-extracts/ (https://mapzen.com/data/metro-extracts/) to inspect the preselected metro area that I downloaded originally. Upon further inspection, the selected area for San Jose, CA is a square that includes all of these represented cities.

Data Overview and Additional Ideas

File sizes

```
san-jose.osm ........... 344 MB
sanjose.db ............. 296 MB
nodes.csv .............. 133 MB
nodes_tags.csv ........ 2.88 MB
ways.csv ................ 13 MB
ways_tags.csv ........... 20 MB
ways_nodes.cv ........... 46 MB
```

Number of nodes

```
sqlite> select count(*) from nodes;

1664111
```

Number of ways

```
sqlite> select count(*) from ways;

226999
```

Number of unique users

```
sqlite> select count(distinct(users.uid))

...> from (select uid from nodes union all select uid from ways) users;


1336
```

Top 10 Popular Cuisines

```
sqlite> select nodes_tags.value, count(*) as num
...> from nodes_tags
...> join (select distinct(id) from nodes_tags where value = 'restaurant') res
...> on nodes_tags.id = res.id
...> where nodes_tags.key = 'cuisine'
...> group by nodes_tags.value
...> order by num desc
...> limit 10;
```

```
vietnamese|70
chinese|65
mexican|60
pizza|53
japanese|43
italian|30
indian|29
american|28
thai|27
sushi|22
```

These results are not surprising at all as the demographic makeup of San Jose is predominantly Asian and Hispanic.

Top 10 Amenities

```
sqlite> select value, count(*) as num
...> from nodes_tags
...> where key = 'amenity'
...> group by value
...> order by num desc
...> limit 10;

restaurant|839
fast_food|421
bench|258
cafe|250
bicycle_parking|186
place_of_worship|185
toilets|158
school|146
bank|125
fuel|123
```

Restaurant Density by Postal Code

```
sqlite> select nodes_tags.value, count(*) as num
...> from nodes_tags
...> join (select distinct(id) from nodes_tags where value = 'restaurant') j
...> on nodes_tags.id = j.id
...> where nodes_tags.key = 'postcode'
...> group by nodes_tags.value
...> order by num desc
...> limit 10;

95014|24
95035|18
95051|12
94086|11
95112|11
94087|8
95128|8
95054|7
95113|7
95123|7
```

From this query we see that the zip code 95014 (Cupertino) has the most restaurants. Overall the amount of restaurants per zip code is small as not all of the restaurants have a postal code listed in the OSM file.

Additional Ideas

It is odd that so little restaurants have their postal code listed. From the query for top 10 amenities we see that there are 839 restaurants. We can perform a qeury to see exactly how many of these restaurants have a listed postal code.

```
sqlite> select count(*) as num
...> from nodes_tags
...> join (select distinct(id) from nodes_tags where value = 'restaurant') j
...> on nodes_tags.id = j.id
...> where nodes_tags.key = 'postcode'
...> order by num desc;
```

186

There is a staggering amount of missing postcodes. My suggestion would be to require users to input a postal code for types of nodes like these. Having this requirement would make the OSM data overall be more complete. Although it is not difficult to add this data into the database by cross referencing the listed address with a database of zip codes.

Conclusion

After auditing the OSM file, it is clear that the San Jose data is incomplete and contains a multitude of errors. For the purposes of this exercise I chose to clean the street names and the postal codes, but there are further areas that the data can be improved. I believe we can further improve this data by auditing other fields such as City name and cuisine type. For example "Vietnamese" and "Pho" can be combined when it comes to cuisine. Even with these preliminary fixes it still represents a great improvement over the existing OSM data.