OpenStreetMap Project
Data Wrangling with MongoDB

Map Area: San Francisco Bay Area, United States

http://metro.teczno.com/#sf-bay-area

1. Problems Encountered in the Map

After initially downloading a large sample size of the SF bay area and running it against a provisional P3OpenStreetMap.py file, I noticed three main problems with the data, which I will discuss in the following order:

- Over-abbreviated street names ("S Tryon St Ste 105")
- Inconsistent postal codes ("95023-9685", "28226")
- "Incorrect" postal codes (*filled with a full address such as "San Juan Bautista, CA 95045"*)

Over-abbreviated Street Names

Once the data was imported to MongoDB, some basic querying revealed street name abbreviations and postal code inconsistencies. I updated all substrings in problematic address strings, such that "Monterey St 402 6t S" becomes "Monterey Street 402 6t South". More examples are in the following:

Powell St → Powell Street
Ashford Cir → Ashford Circle
Fair Oaks Blvd. → Fair Oaks Boulevard
Technology Pkwy → Technology Parkway

S California Ave → South California Avenue
Ruger Ct → Ruger Center

The corresponding code is as follows:

```
if is_street_name(tag): # if match 'addr:street'
  m = street_type_re.search(tag.attrib['v'])
  if m:
     street_type = m.group()
     if street_type not in expected: # if it's not an expected street name
        tag_at_v=update_name(tag.attrib['v'], mapping) # spell it out
```

Postal Codes

Postal code strings posed a different sort of problem, forcing a decision to strip all leading and trailing characters before and after the main 5-digit zip code. This effectually dropped all leading state characters (as in "CA 95045") and 4-digit zip code extensions following a hyphen ("95023-9685"). This 5-digit constriction benefits MongoDB aggregation calls on postal codes. Also, some "incorrect" postal codes ("902 2nd St, San Juan Bautista, CA 95045") surfaced when grouped together with this aggregator:

```
# Sort postcodes by count, descending

    db.char.aggregate([{"$match":{"address.postcode":{"$exists":1}}},
    {"$group":{"_id":"$address.postcode", "count":{"$sum":1}}}, {"$sort":{"count":1}}])
```

Here are some of results, beginning with the highest count:

```
[{'count': 1, '_id': '95610'}, {'count': 1, '_id': '94572'}, {'count': 1, '_id': '95132'},
{'count': 1, '_id': '95121'}, {'count': 1, '_id': '95127'}, {'count': 1, '_id': '95498'},
{'count': 1, '_id': '94937'}, {'count': 1, '_id': '94568'}, {'count': 1, '_id': '95618'},
{'count': 1, '_id': '95211'}, {'count': 1, '_id': '94542'}, {'count': 1, '_id': '94603'},
{'count': 1, '_id': '95023-9685'}, {'count': 1, '_id': '95023-9146'}, {'count': 1, '_id':
'95023-8937'}, {'count': 1, '_id': '95023-9631'}, {'count': 1, '_id': '95045-9728'},
{'count': 1, '_id': '95045-9633'}, {'count': 1, '_id': '95045-9638'}, {'count': 1, '_id':
'95045-9636'},  ..., {'count': 1, '_id': 'San Juan Bautista, CA 95045'}, …
```

To make postcodes consistent and correct, the audit code was added in the following,

```
if is_postcode(tag): #match “addr:postcode”
  m = pc_type_re2.search(tag_at_v)
  if m: # if there is ‘-’
     tag_at_v=tag_at_v[:5] # 95004-9610 --> 95004
  else:
     m = pc_type_re1.search(tag_at_v)
     if m: # if there are alphabets
        tag_at_v=tag_at_v[len(tag_at_v)-5:] # 411 Cole Rd, Aromas, CA 95004  -->  95004
```

leading to:

95023-9685 → 95023
95045-9728 → 95045
San Juan Bautista, CA 95045 → 95045

In addition to the inconsistency of postcodes, there are three more issues I encountered, as follows:

Not complete Node Creation

Once the data was imported to MongoDB, some basic querying revealed incomplete node creation. That is, some nodes do not have “uid” key and value, as opposed to having keys of ["version", "changeset", "timestamp", "user", "uid"].

Large size of the osm file

A GB size of data took a lot of time to manipulate. So, it was better to reduce it down by a factor of 2, based on the code presented in the project. It became much faster even though we missed data to be considered.

Verify JSON

My code was not correct to generate a json file, which I didn’t notice. Somehow, it required a way to verify the correctness of JSON. Although the website jsonlint.com provides the way, it could not be used because of the lage size of it. Eventually, it was able to be modified, ending up getting a correct JSON file.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

sf-bay-area.osm .... 1.45 GB
abbreviated sf-bay-area.osm ……. 736MB
abbreviated sf-bay-area.osm.json ……. 809MB

# Number of documents

```
> db.char.find().count()
3569660
```

# Number of nodes

```
> db.char.find({"type":"node"}).count()
3283945
```

# Number of ways

```
> db.char.find({"type":"way"}).count()
285530
```

# Number of unique users

```
> len(db.char.distinct("created.user"))
2299
```

# Top 1 contributing user

```
> db.char.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":1}])

[{'count': 826838, '_id': 'nmixter'}]
```

# Number of users appearing only once (having 1 post)

```
> db.char.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}},
{"$limit":1}])

[{'num_users': 479, '_id': 1}]
# “_id” represents postcount
```

3. Additional Ideas

Contributor statistics and gamification suggestion

The contributions of users seems incredibly skewed, possibly due to automated versus manual map editing (*the word “bot” appears in some usernames*). Here are some user percentage statistics:

Top user contribution percentage (“'nmixter'”) - 23.16%
Combined top 2 users' contribution (“'nmixter'” and “woodpeck_fixbot”) - 33.94%
Combined Top 10 users contribution - 66.23%

In the context of the OpenStreetMap, if user data were more prominently displayed, perhaps others would take an initiative in submitting more edits to the map. And, if everyone sees that only a handful of power users are creating more than 50% of given map, that might spur the creation of more efficient bots, especially if certain gamification elements were present, such as rewards, badges, or a leaderboard.

Additional data exploration using MongoDB queries

# Top 10 appearing amenities

```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1}}},
{"$group":{"_id":"$amenity","count":{"$sum":1}}}, {"$sort":{"count":-1}},
{"$limit":10}])

[{'count': 4581, '_id': 'parking'}, {'count': 2209, '_id': 'school'}, {'count': 1693, '_id':
'restaurant'}, {'count': 1492, '_id': 'place_of_worship'}, {'count': 712, '_id':
```

```
'fire_hydrant'}, {'count': 627, '_id': 'fast_food'}, {'count': 551, '_id': 'cafe'}, {'count': 536, '_id': 'toilets'}, {'count': 505, '_id': 'bench'}, {'count': 426, '_id': 'fuel'}]
```

# Biggest religion (no surprise here)

```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}}, {"$group":{"_id":"$religion", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}])

[{'count': 1336, '_id': christian}]
```

# Most popular cuisines

```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant", "cuisine":{"$exists":1}}}, {"$group":{"_id":"$cuisine", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":2}])

[{'count': 129, '_id': 'mexican'}, {'count': 87, '_id': 'pizza'}]
```

Conclusion

After this review of the data it’s obvious that the SF area is incomplete, though I believe it has been well cleaned for the purposes of this exercise. It interests me to notice a fair amount of GPS data makes it into OpenStreetMap.org on account of users’ efforts, whether by scripting a map editing bot or otherwise.