

OpenStreetMap Project

Data Wrangling with MongoDB

Kavya Gautam

Map Area: San Jose, California, USA

https://s3.amazonaws.com/metro-extracts.mapzen.com/san-jose_california.osm.bz2

Problem Description:

In this project we are supposed to pick up an area of the world from <https://www.openstreetmap.org> and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the map data. Once the data is audited and cleansed, it should be inserted into mongo DB. This data should be analyzed by running queries in mongo DB. If need be, data can be iteratively cleaned in mongo DB as well.

Solution:

I downloaded San Jose map data from mapzen. I unzipped the tarball and used ElementTree package to parse XML data of San Jose. The main objective was to wrangle the data and transform the shape of the data into a list of dictionaries that look like this:

```
{
  "id": "2406124091",
  "type": "node",
  "visible": "true",
  "created": {
    "version": "2",
    "changeset": "17206049",
    "timestamp": "2013-08-03T16:43:42Z",
    "user": "linuxUser16",
    "uid": "1219059"
  },
  "pos": [41.9757030, -87.6921867],
  "address": {
    "houseNumber": "5157",
    "postcode": "60625",
    "street": "North Lincoln Ave"
  },
  "amenity": "restaurant",
  "cuisine": "mexican",
  "name": "La Cabana De Don Luis",
  "phone": "1 (773)-271-5176"
}
```

For semantic consistency following rules were used for auditing and cleaning:

- Process only 2 types of top level tags: "node" and "way"
- all attributes of "node" and "way" should be turned into regular key/value pairs, except:

- attributes in the CREATED array should be added under a key "created"
- attributes for latitude and longitude should be added to a "pos" array, for use in geospatial indexing.
Make sure the values inside "pos" array are floats and not strings.
- if the second level tag "k" value contains problematic characters, it should be ignored
- if the second level tag "k" value starts with "addr:", it should be added to a dictionary "address"
- if the second level tag "k" value does not start with "addr:", but contains ":",
you can process it in a way that you feel is best. For example, you might split it into
a two-level dictionary like with "addr:", or otherwise convert the ":" to create a
valid key.
- if there is a second ":" that separates the type/direction of a street,
the tag should be ignored, for example:

```
<tag k="addr:housenumber" v="5158"/>
<tag k="addr:street" v="North Lincoln Avenue"/>
<tag k="addr:street:name" v="Lincoln"/>
<tag k="addr:street:prefix" v="North"/>
<tag k="addr:street:type" v="Avenue"/>
<tag k="amenity" v="pharmacy"/>
```

should be turned into:

```
{...
"address": {
  "houenumber": 5158,
  "street": "North Lincoln Avenue"
}
"amenity": "pharmacy",
...
}
```

- for "way" specifically:

```
<nd ref="305896090"/>
<nd ref="1719825889"/>
```

should be turned into

```
"node_refs": ["305896090", "1719825889"]
```

Problems encountered in the map:

During audit, I noticed several issues. First, I removed all objects which contained problem characters using a regular expression match.

```
problemchars = re.compile(r'[\+/\&<>;\ '"\?%#$@\,\.\ \t\r\n]')
```

Also I removed objects which had certain fields abruptly starting with ":". Following were the major issues in the data set:

- Over-abbreviated street names (“Zanker Rd.”)
- Inconsistent postal codes (“95014-2225”, “CA 95110”)
- “Incorrect” postal codes (Zip codes were greater than 5 digits - 951251. Some of them contained just alphabets -CUPERTINO)

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 street types in problematic address strings to form a more consistent naming scheme. For example, “Zanker Rd.” becomes “Zanker Road” and “E Dunne Ave” becomes “E Dunne Avenue”.

Following regular expression is used to get end street type in a street:

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
```

It is then compared with the expected list of street types:

```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road", "Trail",
"Parkway", "Commons", "Way", "Loop", "East", "West", "Terrace", "Expressway", "Plaza", "Hill",
"Highway", "Circle"]
```

If a street type is not present in the expected list, then it is replaced in the original street string based on the below mapping table:

```
mapping = { "St": "Street",
            "St.": "Street",
            "Sq.": "Square",
            "Sq": "Square",
            "Ave": "Avenue",
            "Rd.": "Road",
            "Rd": "Road",
            "Blvd": "Boulevard",
            "Cir": "Circle",
            "Dr": "Drive",
            "Hwy": "Highway"
          }
```

Postal Code related inconsistency and errors:

To solve postal codes with “-” i.e. which specifies range of zip codes, I had to split the postal code using split(“-”) and use the first complete 5 digit code. For example: “95050-5099” gets replaced with “95050”.

If a postal code such as “CA 95050” is present, I use the below snippet to find the zip code which is made up of pure digits. In this example, the cleansed postal code will be 95050.

```
pcs = postcodes[0].split(" ")
for x in pcs:
    n = re.search('[a-zA-Z]', x)
```

Finally, if a postal code is made up of more than 5 digits (typical size of any postal code in USA), then I ignore processing that object.

Overview of the data

Inserting Data into Mongo DB:

Once data was audited and cleansed, I converted XML OSM data file to JSON file. This JSON file is parsed and each individual documents were inserted as documents into a new “examples” database and collection “sanjose”.

Size of Data:

san-jose_california.osm	279.2 MB
san-jose_california.osm.json	317.5 MB

Since the size of data was enormous, I had to batch DB inserts for performance gains. I used mongodbmport for doing bulk insert of json file.

```
os.system("~/mongodb/bin/mongoimport --db examples --collection sanjose --file san-jose_california.osm.json --jsonArray")
```

Doing bulk insert reduced time taken to insert this data set from 4 minutes to 25 seconds.

Interesting facts about San Jose data:

Total number of documents:

```
num_docs = db.sanjose.find().count()
1421903
```

Total number of nodes:

```
num_nodes = db.sanjose.find({"type":"node"}).count()
1256782
```

Total number of ways:

```
num_ways = db.sanjose.find({"type":"way"}).count()
165084
```

Top 10 amenities:

```
top10_amenities = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1}},
    {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
    {"$sort":{"count":-1}},
    {"$limit":10}]]
[{u'_id': u'parking', u'count': 1712},
{u'_id': u'restaurant', u'count': 706},
{u'_id': u'school', u'count': 509},
{u'_id': u'fast_food', u'count': 404},
{u'_id': u'place_of_worship', u'count': 305},
```

```
{u'_id': u'fuel', u'count': 200},
{u'_id': u'cafe', u'count': 188},
{u'_id': u'bench', u'count': 181},
{u'_id': u'toilets', u'count': 168},
{u'_id': u'bicycle_parking', u'count': 163}}
```

Ethnicity distribution using Cuisine Count:

Top 2 Popular cuisines:

```
popular_cuisines = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1},
    "cuisine":{"$exists":1},
    "amenity":"restaurant"}},
    {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
    {"$sort":{"count":-1}}, {"$limit":2})]]
[{u'_id': u'mexican', u'count': 63}, {u'_id': u'vietnamese', u'count': 48}]
```

Top 30 Popular cuisines:

```
popular_cuisines = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1},
    "cuisine":{"$exists":1},
    "amenity":"restaurant"}},
    {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
    {"$sort":{"count":-1}}, {"$limit":30})]]
[{u'_id': u'mexican', u'count': 63},
{u'_id': u'vietnamese', u'count': 48},
{u'_id': u'chinese', u'count': 47},
{u'_id': u'pizza', u'count': 44},
{u'_id': u'japanese', u'count': 35},
{u'_id': u'indian', u'count': 26},
{u'_id': u'american', u'count': 25},
{u'_id': u'italian', u'count': 25},
{u'_id': u'sushi', u'count': 17},
{u'_id': u'thai', u'count': 16},
{u'_id': u'sandwich', u'count': 12},
{u'_id': u'burger', u'count': 12},
{u'_id': u'steak_house', u'count': 10},
{u'_id': u'regional', u'count': 8},
{u'_id': u'mediterranean', u'count': 7},
{u'_id': u'greek', u'count': 6},
{u'_id': u'seafood', u'count': 5},
{u'_id': u'korean', u'count': 4},
{u'_id': u'barbecue', u'count': 4},
{u'_id': u'asian', u'count': 4},
{u'_id': u'chicken', u'count': 4},
{u'_id': u'hawaiian', u'count': 3},
{u'_id': u'breakfast', u'count': 2},
{u'_id': u'Vietnamese', u'count': 2},
{u'_id': u'donuts', u'count': 2},
{u'_id': u'persian', u'count': 2},
{u'_id': u'international', u'count': 2},
{u'_id': u'malaysian', u'count': 2},
{u'_id': u'coffee_shop', u'count': 2},
```

```
{u'_id': u'ice_cream', u'count': 2},
{u'_id': u'french', u'count': 2},
{u'_id': u'ethiopian', u'count': 2},
{u'_id': u'spanish', u'count': 2},
{u'_id': u'salvadoran', u'count': 1},
{u'_id': u'afghan', u'count': 1},
{u'_id': u'cuban', u'count': 1},
{u'_id': u'buffet', u'count': 1},
{u'_id': u'texmex', u'count': 1},
{u'_id': u'brazilian', u'count': 1},
{u'_id': u'jamaican', u'count': 1},
```

We can see that there are more Mexican, Chinese, Japanese and Indian restaurants than American restaurants in San Jose. This suggests that people of corresponding ethnicity are more than Americans in San Jose. We are also able to see that Brazilians and Jamaicans cuisines are very few, suggesting that the corresponding ethnicity might be very scarce in San Jose.

Auditing and Cleansing Data in Mongo DB:

I wanted to find all Indian cuisines in the area out of curiosity.

Total Indian cuisines:

```
db.sanjose.find({"cuisine":"indian", "amenity":"restaurant"}).count()
```

26

All Indian cuisines:

```
indian_cuisines = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1},
                                                         "cuisine":"indian",
                                                         "amenity":"restaurant"}},
                                                         {"$group":{"_id":"$name", "count":{"$sum":1}}},
                                                         {"$sort":{"count":-1}}]]]
```

```
{u'_id': u'Swaad Indian Cuisine', u'count': 1},
{u'_id': u'Ananda Bhavan', u'count': 1},
{u'_id': u'Tandoori Oven', u'count': 1},
{u'_id': u'8Elements', u'count': 1},
{u'_id': u'Silver Spoon', u'count': 1},
{u'_id': u'Dosa & Curry Cafe', u'count': 1},
{u'_id': u'Sagar Sweets', u'count': 1},
{u'_id': u'Turmeric', u'count': 1},
{u'_id': u'Rasam\u2019s', u'count': 1},
{u'_id': u'Spice Hut', u'count': 1},
{u'_id': u'Athidhi Indian Cuisine', u'count': 1},
{u'_id': u'Madhuban Indian Cuisine', u'count': 1},
{u'_id': u'Curry Bhavan', u'count': 1},
{u'_id': u'Sneha', u'count': 1},
{u'_id': u'Chaat House', u'count': 1},
{u'_id': u'Madura Indian restaurant', u'count': 1},
{u'_id': u'Chaat Bhavan', u'count': 1},
{u'_id': u'Punjab Cafe', u'count': 1},
{u'_id': u'Amber India', u'count': 1},
```

```
{u'_id': u'L&L Hawaiian BBQ', u'count': 1},
{u'_id': u'Tandoori Bites', u'count': 1},
{u'_id': u'Kaati Fresh', u'count': 1},
{u'_id': u'Panchavati', u'count': 1},
{u'_id': u'Komala Vilas', u'count': 1},
{u'_id': u'Aasna, Melange of India', u'count': 1},
{u'_id': u'Tandoori Cafe', u'count': 1}]
```

Iterative Data Cleaning using Mongo DB:

```
ll = db.sanjose.find_one({"name": "L&L Hawaiian BBQ"})
ll['cuisine'] = "American"
db.sanjose.save(ll)
```

Total Indian cuisines after update:

```
db.sanjose.find({"cuisine":"indian", "amenity":"restaurant"}).count()
25
```

Other ideas about the dataset

Additional data exploration using MongoDB queries:

I carried out the below queries to gain more insight into San Jose data set.

Total number of hospitals:

```
num_hospitals = db.sanjose.find({"amenity":"hospital"}).count()
22
```

Total number of schools:

```
num_schools = db.sanjose.find({"amenity":"school"}).count()
509
```

Total number of universities:

```
num_univ = db.sanjose.find({"amenity":"university"}).count()
9
```

Top contributing user:

```
top_user = [doc for doc in db.sanjose.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}},
{"$sort":{"count":-1}},
{"$limit":1}}])]
[{u'_id': u'nmixer', u'count': 286454}]
```

Number of 1-time users:

```
user_1time = [doc for doc in db.sanjose.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}},
{"$sort":{"_id":1}}, {"$limit":1}}])]
[{u'_id': 1, u'num_users': 192}]
```

Zip code used the most:

```
pc_sorted = [doc for doc in db.sanjose.aggregate([{"$match":{"address.postcode":{"$exists":1}}},
{"$group":{"_id":"$address.postcode", "count":{"$sum":1}}},
{"$sort":{"count":-1}}]]
```

```
[{'u_id': 'u'95014', 'u_count': 9862},
{'u_id': 'u'95050', 'u_count': 40},
{'u_id': 'u'95128', 'u_count': 25},
{'u_id': 'u'94087', 'u_count': 21},
{'u_id': 'u'95037', 'u_count': 15},
{'u_id': 'u'94086', 'u_count': 14},
{'u_id': 'u'95110', 'u_count': 13},
{'u_id': 'u'95125', 'u_count': 12},
{'u_id': 'u'95070', 'u_count': 10},
{'u_id': 'u'95129', 'u_count': 9},
{'u_id': 'u'95051', 'u_count': 8},
{'u_id': 'u'95131', 'u_count': 8},
{'u_id': 'u'95123', 'u_count': 8},
{'u_id': 'u'95118', 'u_count': 7},
{'u_id': 'u'95054', 'u_count': 5},
{'u_id': 'u'95112', 'u_count': 5},
{'u_id': 'u'94085', 'u_count': 5},
{'u_id': 'u'95008', 'u_count': 5},
{'u_id': 'u'95032', 'u_count': 4},
{'u_id': 'u'95126', 'u_count': 3},
{'u_id': 'u'95113', 'u_count': 3},
{'u_id': 'u'95120', 'u_count': 3},
{'u_id': 'u'95002', 'u_count': 3},
{'u_id': 'u'95035', 'u_count': 3},
{'u_id': 'u'95124', 'u_count': 3},
{'u_id': 'u'95134', 'u_count': 2},
{'u_id': 'u'94089', 'u_count': 2},
{'u_id': 'u'95132', 'u_count': 2},
{'u_id': 'u'95030', 'u_count': 2},
{'u_id': 'u'95136', 'u_count': 1},
{'u_id': 'u'95127', 'u_count': 1},
{'u_id': 'u'94807', 'u_count': 1},
{'u_id': 'u'95013', 'u_count': 1},
{'u_id': 'u'95119', 'u_count': 1},
{'u_id': 'u'95138', 'u_count': 1},
{'u_id': 'u'95111', 'u_count': 1},
{'u_id': 'u'95914', 'u_count': 1}]
```

List of available universities:

```
all_univ = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1},
"name":{"$exists":1}, "amenity":"university"}},
{"$group":{"_id":"$name", "count":{"$sum":1}}},
{"$sort":{"count":-1}}]]
```

```
[{'u_id': 'u'Santa Clara University', 'u_count': 1},
```



```
{u'_id': u'National Hispanic University', u'count': 1},
{u'_id': u'Bay Area Campus', u'count': 1},
{u'_id': u'Medical Center - University of E-W Medicine', u'count': 1},
{u'_id': u'Five Branches University', u'count': 1},
{u'_id': u'San Jos\xe9 State University Main Campus', u'count': 1},
{u'_id': u'San Jos\xe9 State University South Campus', u'count': 1},
{u'_id': u'Lincoln Law School of San Jose', u'count': 1}]
```

Biggest religion:

```
biggest_religion = [doc for doc in db.sanjose.aggregate([{"$match":{"amenity":{"$exists":1},
    "amenity":"place_of_worship"}},
    {"$group":{"_id":"$religion", "count":{"$sum":1}}},
    {"$sort":{"count":-1}, {"$limit":1}}]]
[{u'_id': u'christian', u'count': 264}]
```

Conclusion:

This project gave me very good insight into Mongo DB and data wrangling techniques. After auditing and cleaning OpenStreetMap data, I feel that San Jose data is cleansed enough for the purposes of this project. However, there are still many areas which can be cleaned in this data set. Also, it will benefit everyone if we data analysts start updating OpenStreetMap with cleansed data.