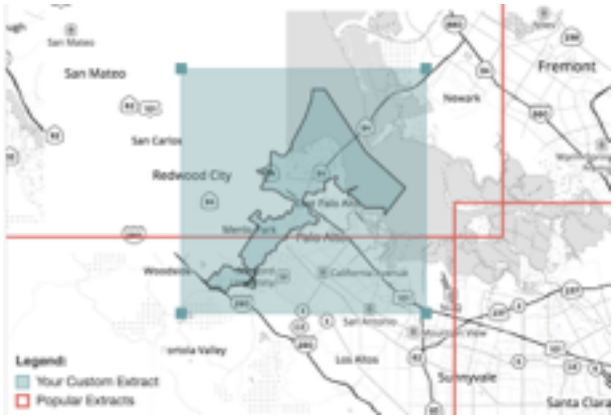
OpenStreetMap Sample Project
Data Wrangling with MongoDB
Tony Nguyen

Menlo Park, California (and others), United States
https://www.openstreetmap.org/relation/1544957#map=12/37.4758/-122.1553
https://mapzen.com/data/metro-extracts/



I used MapZen to extract the OSM data used in this report. The green highlighted area in the picture to the left depicts the area of data extracted. Notice that Menlo Park and some of it's surrounding areas were extracted.

Problems Encountered in the Map

1) Data sample size
I wanted to do a study of Menlo Park, California , and after downloading the OSM (named Menlo OSM) file I noticed that the file contained data from other cities as well. My first step was to extract data that only contained information about Menlo Park by using ElementTree's iterfind function and use that data as the sample. However after reviewing and analyzing the Menlo Park sample, I discovered the data was too small. There was insufficient data to fully grasp what kind of errors the initial OSM file contains.

Two errors was not enough for me. I was forced to generate a sample of the initial OSM file, which contains data from other cities, instead of using a sample that just contained data from Menlo Park (titled sample OSM).

Using what I learned from the case study, I organized the data so I can easily see what kind of information needed to be fixed.

2) Titling City Names
This is the count of tags grouped by cities. I noticed that there could be some cities where their names was not titled.

{'Redwood City': 346, 'Menlo Park': 2, 'Mountain View': 31, 'Stanford': 5, 'menlo park': 1, 'Palo Alto': 368, 'East Palo Alto': 1}

- ‘menlo park’ needs to be capitalized

3) City Abbreviation and Odd Streets
This is the count of tags grouped by street names. The sample did not contain any abbreviation of streets, but I did write code to fix it just in case. I also queried any streets that looked off to me, and cross referenced any unexpected streets with Google Maps.

{'Real': 13, 'Lane': 24, 'Boulevard': 5, 'Court': 31, 'Center': 1, 'Alameda': 2, 'Drive': 63, 'Escarpado': 1, 'Walk': 2, 'Vista': 1, 'Mall': 1, 'Street': 190, 'Place': 14, 'Way': 50, 'Broadway': 7, 'Parkway': 1, 'Las': 1, 'Avenue': 256, 'Circle': 8, 'Road': 51, 'Row': 1}

- abbreviation of avenue (ave. VS avenue)
- some streets end incorrectly
    - Escarpado = El Escarpado Avenue
    - Las = Alameda de las Pulgas
    - Alameda = Alameda de las Pulgas

With street name that contained no sweet type (eg: Escarpado) I had to go in manually and correct those streets because there was no way of bulk editing them. I corrected any abnormalities in the sample ism, but there could be more odd street type endings in the Main OSM file.

4) Postal Codes
There are some postal codes that displayed the full 9 digit code, while a majority displayed 5 digits. (sample OSM)

- 94301-2019
- CA91583 (I did not find any postal codes of this type, but created some codes to fix it just in case)

5) Missing Data
I wanted to double check the data I extracted into MongoDB and some codes I used in python. For example, after generating a count for total cities were in the sample osm, there was a slight discrepancy in the data.

Python {Mountain View : 3, Redwood City : 35, Stanford: 1, Palo Alto: 38}
Mongo {Mountain View : 3, Redwood City’: 34,'Stanford: 1, Palo Alto: 37}
city_count = [{‘$group':{"_id":"$address.city",'count':{'$sum':1}}},{'$sort':{'count':-1}}]

Using pymongo’s find and projection functions and ElementTree’s iterfind function I compiled a list of all id’s and compared searched for any thing missing. It turns out that tags nested within nodes were extracted into the json file, but tags texted within ways were not. (I also checked this with the case study, and the same thing happens). For

‘{changeset': '18166706',

'id': 240420866,
'input_by': {'uid': '169004', 'user': 'oldtopos'},
'timestamp': '2013-10-03T19:23:26Z',
'type': 'way',
'version': 1.0}

This way has a tag nested within it, but is not extracted when converted into json.

Stats of the data:

Menlo.osm - 181.7 mb
Menlo.json - 186.2 mb

Number of documents
mnelo.find().count()
Menlo json - 928235

Number of Nodes/Ways

Menlo json
menlo.find({'type':'node'}).count() and menlo.find({'type':'way'}).count()
nodes - 832583
ways - 95652

Number of distinct users
len(menlo.distinct("input_by.user"))
668

Top 10 contributors
most_posts = [{'$group':{"_id":"$input_by.user",'count':{'$sum':1}}},{'$sort':{'count':-1}},
{"$limit":10}]

{u'_id': u'oldtopos', u'count': 227856}
{u'_id': u'RichRico', u'count': 139563}
{u'_id': u'ediyes', u'count': 104140}
{u'_id': u'karitotp', u'count': 79897}
{u'_id': u'calfarome', u'count': 76913}
{u'_id': u'samely', u'count': 74318}
{u'_id': u'dannykath', u'count': 57462}
{u'_id': u'andygol', u'count': 13259}
{u'_id': u'Harry Cutts', u'count': 10952}
{u'_id': u'KindredCoda', u'count': 10861}

Additional data exploration using MongoDB queries

Top 10 Counties:
county = [{"$group":{"_id":"$details.county", “count”:{"$sum":1}}},{'$sort':{'count':-1}}
{"$limit":10}]

{u'_id': None, u'count': 923243}
{u'_id': u'San Mateo, CA', u'count': 2891}
{u'_id': u'Santa Clara, CA', u'count': 2006}
{u'_id': u'Alameda de las Pulgas', u'count': 48}
{u'_id': u'San Mateo, CA;Santa Clara, CA', u'count': 21}
{u'_id': u'San Mateo, CA:Santa Clara, CA', u'count': 13}
{u'_id': u'San Mateo, CA; Santa Clara, CA', u'count': 13}

It is interesting to note that there are places with more than one county. This could be because some streets run through more then one county. However 'Alameda de las Pulgas’ is not a count yet there are 48 elements that lists 'Alameda de las Pulgas’ as a county.

Additional Ideas

Separate field for street types
Problem:
There are many different and odd street names that are not typical. For example, Las or Alameda are not typical and would be difficult to distinguish the element as a street type or an error in data entry.

Solution: Separating street names and street street types into two different fields.

Benefit 1: Allows users to preform better queries. The option to query all streets of a certain type will result in more insightful data to be returned.
Benefit 2: Create better accuracy when inputting the data. One problem that occurred for me was a street type, “Las”. In actuality the street was Alameda de las Pulgas. Having two separate fields will ensure that a street type will always be inputed
Benefit 3: Faster auditing. Instead of creating complicated code to look for abnormalities, Users can just use the street types field.

Issue 1: Implementing this change. Since there is already a standard format it would be timing consuming to implement this change for the entire database.
Issue 2: Update program codes that use the old format. Any program or software would need an update to accommodate the changes

Improvement on unincorporated areas or trends

Problem: There is no way to track and manage unincorporated areas or trends. For example, Stanford is technically not a city, but it's not wrong to have some elements label Stanford as a city. Another example would be stores in Malls. All these stores would have the same address of the mall, and it could be difficult to distinguish one from another.

Solution: Creating primary and secondary address fields. Primary will contain the standard address while secondary will contain everything else; trends, unincorporated areas, etc.

> **Benefit**: This will allow the option have having all the data present for a particular place.
>
> **Issue**: It would be difficult to query. Not all places require a secondary address and it would difficulty to know what's there in the secondary address, unless you knew before hand that it was already there.

Conclusion

There was a lot to clean up and I did my best to amend any irregularities found in the data. However there are still a lot more cleaning that could be done. Even after my own clean up, I found a few elements that were entered in erroneously. There should be some type audits before any data goes live in the Open Street Maps Database.

Resources

Mongo DB - https://docs.mongodb.com/manual/aggregation/
Stack Overflow - searched multiple queries
Element Tree python doc - https://docs.python.org/2/library/xml.etree.elementtree.html

Udacity's Data Analyst Course and Resources
Scripts taken from the quizzes that were used for my project.

```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane",

"Road",


"Trail", "Parkway", "Commons"]

# typical street abbreviations


mapping = {"St.": "Street",  "St": "Street", "Rd": "Road", "Rd.": "Road", "Ave": "Avenue",

"Ave.": "Avenue"}
```

```
# some regular expression searches to look for anomalies (taken from the quiz) lower =
re.compile(r'^([a-z]|_)*$') # finds all lower case keys
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$') # finds all keys with a colon
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t \r\n]') # finds any problem
characters

street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE) # find the street types
odd_keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
```