Open Street Map Project - Data Wrangling with MongoDB

Adam Carney

Map Area: Phoenix, Arizona, United States
https://mapzen.com/data/metro-extracts/metro/phoenix_arizona/

Problems Encountered

After a review of the data I noticed the following issues I will discuss. They are:

- City Names
- House numbers were absent

City Names

82 cities show up in the Phoenix data (I had created a full mongoDb version to investigate initially). There are some names misspelled, case reversed (ex: tEMPE), and included the state code in the city name (ex: Mesa, AZ). Additionally, there are city names that don't actually exist in the Phoenix area (ex: Seattle, San Diego).

Searching my initial mongoDB instance: { "$match": {"address.city": {"$exists":1}}},{ "$group": {"_id": "$address.city", "count": { "$sum": 1}}},{"$sort" : {"count" : -1}}

I found the following issues, which I updated the following during the conversion for my final data extract:
MEsa -> Mesa
tempe -> Tempe
Paradise Valley, AZ -> Paradise Valley
chandler -> Chandler
CHANDLER -> Chandler
Mesa, AZ -> Mesa
tEMPE -> Tempe
casa Grande -> Casa Grande
sun City West -> Sun City West
peoria -> Peoria
SanTan Valley -> San Tan Valley
scottsdale -> Scottsdale
mesa -> Mesa

Street Numbers

When investigating street data, the street names were very clean. When printing out the first 100 records, I noticed counts were greater than one. This appears to be because the house number was not filled out. I would expect in a metropolitan area that there would be unique street addresses. This is not true “across the board” as rural areas in the United States will have locations that are not numbered (I know this from my experience in a rural telecommunications company).

{"$match": {"address": {"$exists":1}}},{ "$group": {"_id": "$address", "count": { "$sum": 1}}},{"$sort" : {"count" : -1}}, {"$limit": 100}

Data Investigation

This section contains some basic statistics about the data loaded into MongoDB.

File Sizes

phoenix_arizona.osm: 667.3 MB
phoenix_arizona.osm.json: 746.2 MB

Number of documents
db.osm.find().count()
Total collection: 3242454

Number of Unique Users (I used python length - see code)
Total unique users: 1357

Number of nodes
db.osm.find({"type":"node"}).count()
Total Nodes: 2872267

Number of ways
db.osm.find({"type":"way"}).count()
Total Ways: 370002

Additional Ideas

City Validation

There were some issues in the city name that I believe could be easily avoided by doing some validation, especially in more metropolitan areas. When entering a tag that is an “address.city” I would hope something along the lines below could occur:

- Create a polygonal validation that takes the latitude and longitude of the point being entered and returns/validates the city that is being entered, that can be overwritten if necessary.
- Add a new tag that if the return from bullet one is null, it has an “not validated” tag that is added, so that as you are parsing the data, you can react to data that is not validated more systematically.
- Use USPS street address API (https://www.usps.com/business/web-tools-apis/address-information-api.htm) to validate city name.

Benefits:

- Increased data quality by validating records

- Tags to allow users of the data to decide whether to filter un-validated data or keep it in a more systematic manner.

Potential Issues:

- Services could be down, resulting in confusion when entering data that you may know is accurate, but getting “validation issues”.
- When reviewing/validating areas that are not metropolitan or densely populated the quality of data from the external services may diminish, resulting in either false positives or negatives (this is the reason for suggesting the override option, but still doesn’t completely solve data quality issues).

Street Number/Street Address Validation

There were some issues that occurred in the street address that I believe could have also been avoided, especially in more metropolitan areas.

- Call an API that can do delivery point validation (DPV) or address search against the USPS and allow selection of address if one is not matched, that can be overwritten if necessary.
- Add a new tag that if the return from bullet one is null, it has a “not validated” tag that is added, so that as you are parsing the data, you can react to data that is not validated more systematically.

Benefits:

- Increased data quality by validating records
- Tags to allow users of the data to decide whether to filter un-validated data or keep it in a more systematic manner.

Potential Issues:

- Services could be down, resulting in confusion when entering data that you may know is accurate, but getting “validation issues”.
- When reviewing/validating areas that are not metropolitan or densely populated the quality of data from the external services may diminish, resulting in either false positives or negatives (this is the reason for suggesting the override option, but still doesn’t completely solve data quality issues).
- One of the biggest ones with the suggestion I mentioned, is that some of this only works in the United States, while Open Street Map spans the globe.

Additional Data Investigation

Amenities

Top amenity appears to be parking. The second most was schools. It is interesting that parking is 3.55 times higher than the second amenity, but this makes sense since something like parking has as relationship to amenities (in other words, schools, fast food, places of worship, etc have to have parking). The top 10 in descending order are:

{"$match": {"amenity": {"$exists":1}}},{ "$group": {"_id": "$amenity", "count": { "$sum": 1}}},{"$sort" : {"count" : -1}}, {"$limit": 10}

{u'_id': u'parking', u'count': 4076},
{u'_id': u'school', u'count': 1147},
{u'_id': u'fast_food', u'count': 1067},
{u'_id': u'place_of_worship', u'count': 1019},
{u'_id': u'restaurant', u'count': 955},
{u'_id': u'fuel', u'count': 835},
{u'_id': u'parking_space', u'count': 675},
{u'_id': u'bench', u'count': 594},
{u'_id': u'shelter', u'count': 516},
{u'_id': u'swimming_pool', u'count': 406}

Top 20 users

The top 20 users make up 79% of the total contributions for Phoenix. Top 20 contributors are:

Get top 20 users:
{"$group": {"_id": "$created.user","count": { "$sum": 1 }}},{"$sort" : {"count" : -1}}, { "$limit" : 20}

Total in collection:
db.osm.find().count()

Results
{u'_id': u'Dr Kludge', u'count': 1109431},
{u'_id': u'TheDutchMan13', u'count': 334008},
{u'_id': u'AJ Riley', u'count': 200377},
{u'_id': u'Adam Martin', u'count': 182611},
{u'_id': u'tomthepom', u'count': 114743},
{u'_id': u'adenium', u'count': 99461},
{u'_id': u'CartoCrazy', u'count': 69805},
{u'_id': u'woodpeck_fixbot', u'count': 60406},
{u'_id': u'kghazi', u'count': 59626},
{u'_id': u'namannik', u'count': 42721},
{u'_id': u'AZ_Hiker', u'count': 38862},
{u'_id': u'jfuredy', u'count': 36451},
{u'_id': u'nmixter', u'count': 35163},
{u'_id': u'Alan Bragg', u'count': 30227},
{u'_id': u'Bike Mapper', u'count': 27094},
{u'_id': u'newantt', u'count': 26487},
{u'_id': u'F_H_Howler', u'count': 22842},
{u'_id': u'Chris Lawrence', u'count': 20938},
{u'_id': u'woodpeck_repair', u'count': 20546},
{u'_id': u'Rub21', u'count': 20170}

Total Top 20 user contribution: 2551969

Total collection: 3242454