OpenStreetMap Project in Data Wrangling with MongoDB

Name: Anna Li

Email: annax.li10@gmail.com

Map Area: Auckland, New Zealand

URL: http://www.openstreetmap.org/export#map=12/-36.8612/174.7839

I chose my hometown, Auckland, New Zealand, for project 2 because I grew up in this city, and I know the area well. New Zealand is also far away from the rest of the world, so I want to dedicate my efforts to improve the quality of the Auckland map.

I selected an area to wrangle that includes the central Auckland city and the greater suburban regions, such the North Shore (Takapuna, Birkenhead, and Devonport), all the way West to Henderson and Te Atau and as far South as Mangere Bridge. I have friends living in all of these areas

I downloaded the data in two steps: 1. I found the area I was interested in using OpenStreetMap (http://www.openstreetmap.org/export#map=12/-36.8612/174.7839) and made sure the dataset was over 50MB, 2. I ran a mapping query from Overpass API (http://overpass-api.de/query_form.html) to download the xml data with metadata:

```
(node(-36.9469,174.6243,-36.7755,174.94355);<;);out meta;
```

Table of Contents

Problems in the OSM Auckland Data

After downloading and auditing the original dataset, I discovered three problems in the data set that required wrangling and cleaning. The three problems:

- Lack of standardization in street types (e.g. 'Queen St.' and 'Queen st')
- Incorrect values in fields (e.g. 'Newmarket' suburb is misplaced in the street)
- Missing values (city was often missing from the address)

Standardization

Inconsistent street types

Using the aklaudit.py script, which is modified from the auditing exercise in lesson 6, I discovered inconsistent street types that required cleaning. For example, "Queen St." and

"Queen st" needed to be standardized to "Queen Street". I updated inconsistent street types to match the standardized format and removed leading and trailing whitespaces. Here is an example of the street type transformation:

```
Gillies Ave => Gillies Avenue
Erson Ave => Erson Avenue
Queen st => Queen Street
Queen St. => Queen Street
 New North Rd => New North Road
```

Note: New Zealand streets frequently have the street direction appended at the end of the street name, such as "Customs Street West". I decided not to remove the direction because it is meaningful. This happens because very long streets stretch across the city and "90 Green Lane East" is in a different suburb (Remuera) than "90 Green Lane West" (Epsom).

After importing the JSON-formatted data into MongoDB, I wrote queries with the pymongo connector to retrieve the street names and cities from the Auckland data.

```
db.auckland.aggregate( [{'$match': {'address.street': {'$exists':
1}}},'$group': {'_id': '$address.street','count': { '$sum': 1} }},
{'$sort': {'count': -1}}, {'$limit': 100}] )
```

I was able to look at the different street names to find incorrect values. For example, "Hurstmere" was missing "Road" and "Waverley" was missing "Avenue" (I fixed this in aklaudit.py). I noticed some streets have unusual names and types, such as "The Strand". But this is a correct street name (The Strand is an arcade-style square in the Auckland city center so its street type is an unusual exception) and does not require cleaning.

Incorrect values in fields

City values frequently contain additional data like house number and street

After exploring the addresses, I audited the city data to determine what kinds of inconsistent formatting and values existed, and therefore what needed cleaning. I modified the MongoDB aggregation pipeline query for street to work on city:

```
db.auckland.aggregate( {'$match': {'address.city': {'$exists':
1}}}, {'$group': {'_id': '$address.city', 'count': { '$sum': 1} }},
{'$sort': {'count': -1}}] )
```

The city field had two main problems:

- Inconsistent or incorrect spellings of "Auckland", such as "auckland", "1 Auckland", and "Auckalnd"
- Most of the city values were suburbs in Auckland, such as "Parnell", "Takapuna", and "Morningside". Sometimes there was a suburb concatenated with the city, such as "Birkenhead, Auckland" and "New Lynn, Auckland".

The city 'Auckland' appears correctly 460 times in the OSM data set. There are 28 instances where it requires cleaning are listed below (city and number of occurrences):

'Waterview'	1	'North Shore':	3	'Kelston, Auckland'	1	'Takapuna'	5
'auckland'	1	'Royal Oak'	1	'Birkenhead, Auckland'	1	'Mt Wellington'	1
'Waitemata'	5	'Morningside'	2	'New Lynn, Auckland'	3	'Auckalnd'	1
'St Heliers'	1	'1 Auckland'	1	'Glen Eden'	1		

I fixed the dirty city data by setting all city values to “Auckland” and extracting a suburb with the help of regular expressions (see the script aklcitycleaning.py).

```
suburb_re = re.compile(r'^([a-zA-Z ]*)(,|$)')
auck_re = re.compile(r'^([Auckland])*$')
```

If there were a suburb in the city field, I saved it as a value to the key “suburb” in the JSON data. I hardcoded all the city keys with the value “Auckland” deliberately because I had picked the Auckland metro area so I was confident that all the city values should be “Auckland”.

A similar but less prevalent problem existed in the street field. After standardizing the street types, I discovered a small number of streets contained the house number.

```
"address": {
    "city": "Auckland",
    "street": "161 Halsey Street",
    "housename": "Viaduct Events Centre",
    "postcode": "1010",
    "housenumber": "161"
  }
```

I decided to fix the “street “ value by removing the house number from the street. There was no need to extract the house number and create a new key-value pair like what I did for city and suburb because the value of ‘housenumber’ already existed. Here is the regular expression I used with a re.sub() command to remove the house number from the street:

```
housenumber_re = re.compile(r'(^[0-9 ]+)')
```

Missing values

City is missing from address data that has at least a street

Realizing that many city values were actually suburbs made me wonder how many city fields were missing data. I wrote a query to uncover how many address documents that had at least a street were missing a city:

```
db.auckland.aggregate ( [{'$match': {'address': {'$exists': 1}}},
{'$match': {'address.street': {'$exists': 1}}}, {'$match':
{'address.city': {'$exists': False}}}, {'$group': {'_id':
'$address.city','count': {'$sum': 1}}}] )
```

The result showed there were 801 address documents missing a city.

```
{u'ok': 1.0, u'result': [{u'_id': None, u'count': 801}]}
```

The way I cleaned the suburb data also allowed me to take care of adding a missing city to the address document since I hardcoded every address document with a key-value pair of {‘city’: ‘Auckland’}. Therefore, I filled in the 801 address documents missing a city with the key-value pair {“city”: “Auckland”}.

Overview of OSM Auckland Data

This section provides statistics about the Auckland dataset and the MongoDB queries used to retrieve these statistics.

Data files

OSM Data	File Size	Description
auckland.osm	114.6 MB	(original Auckland dataset from OpenStreetMap)
osm-auckland.json	161.3 MB	(cleansed dataset converted to JSON for MongoDB)

Data Wrangling scripts	Description
aklstreetaudit.py	Audits and standardizes street types and names
aklcitycleaning.py	Audits and standardizes city data and extracts suburb
akldata.py	Converts XML to JSON and performs final cleaning
aklMongoAddressQueries.py	Exploratory data analysis in MongoDB for cleaning
aklMongoExploreDataset.py	Queries for analyzing data after importing clean dataset in MongoDB
aklMongoAddtlAnalysis.py	Additional analysis for interesting project idea

Command line function to import OSM Auckland json data into MongoDB

mongoimport --db osm --collection auckland --file auckland.json –jsonArray

Descriptive statistics on Auckland Dataset

Exploratory analysis in Mongo shell

Number of documents

```
> db.auckland.count()
```
Output: 586017

Number of nodes

```
> db.auckland.find({"type":"node"}).count()
```
Output: 515431

Number of ways

```
> db.auckland.find({"type":"way"}).count()
```
Output: 70586

Exploratory analysis in using Pymongo-based queries

Number of unique users

```
db.auckland.aggregate ([{'$group': {'_id': '$created.user'}},
        {'$group': {'_id': 'Users', 'count': {'$sum': 1}}}])
```
Output: 396

Top 3 contributors to the Auckland dataset

```
db.auckland.aggregate ( [{'$group': {'_id': '$created.user',
'count': {'$sum': 1}}},{'$sort': {'count': -1}},{'$limit': 3}] )
```
Output (user, number of contributions):
1. Rudy355 (187094), 2. Robert Ancell (159313), 3. Myfanwy (124607)

Most common shop facilities

```
db.auckland.aggregate([{'$match': {'shop': {'$exists': 1}}},
{'$group': {'_id': '$shop', 'count': {'$sum': 1}}}, {'$sort':
{'count': -1}}])
```
Output (shop, count): 1. Supermarket (73), 2. Convenience (63), 3. Bakery (27),

4. alcohol (26), 5. Hairdresser (21), 6. Clothes (18)…

Top 5 sports facilities in Auckland (surprisingly, rugby is not number 1)

```
db.auckland.aggregate( [{'$match': {'sport': {'$exists': 1}}},
    {'$group': {'_id': '$sport', 'count': {'$sum': 1}}},
    {'$sort': {'count': -1}}, {'$limit': 5} ])
```

Output (sport, count):

1. *Tennis (313), 2. Bowls (124), 3. Swimming (101), 4. Rugby (55), 4. Cricket (55)*

Information about a series of sport centers called Les Mills

```
gyms = db.auckland.find({'name': 'Les Mills'})
for g in gyms:
   pp.pprint(g)
```

Output: { 'building': 'yes', 'leisure': 'sports_centre', 'name': 'Les Mills', 'id': u'44283778', … (Output was condensed for space)

Most common types of buildings in Auckland

```
db.auckland.aggregate([{'$match': {'building': {'$exists': 1}}},
{'$group': {'_id': '$building', 'count': {'$sum': 1}}},
{'$sort': {'count': -1}}])
```

Output (building type, count): 'yes' (16389), residential (15971), house (282), commercial (187), retail (161) …

Potential Use Case for Dataset

Business Development and Market Research

My friend Phillip Mills owns a successful chain of sport centers and gyms in Auckland. With 11 gyms in New Zealand, 5 of which are in Auckland, he told me he wants to expand his business. The OSM Auckland data would be a great addition to the market research he was already conducting about where to build the gyms. For example, the OSM data shows which sports facilities are the most common in Auckland (tennis, bowls, and swimming) and where they are located based on the latitude and longitude data.

Phillip told me he considers many factors when opening a sports center, including where his competitors are, where office and commercial buildings are (his gyms target a higher-end market), whether there are retail shops around, what highways are nearby so people have easy access to the sports center, and whether there is an existing parking lot or an open space to build a new parking lot.

The OSM Auckland dataset provides a rich dataset to begin answering many of the above questions. Combined with market research, this geographic data could help the Phillip decide on where to build the new facilities.

The biggest downside to the current dataset is the lack of information on existing Les Mills gyms. Querying the data only returned one Les Mills gym in New Lynn. Although the OSM data provides insightful information that could help Phillip make business decisions, improving the data by enriching it with more information on Les Mills gyms and his competitors would greatly increase the benefits of the dataset. Other important information the dataset lacks includes demographic information and population data, which would boost the impact of Phillip's analyses and his conclusions from the data.

Conclusion

The OSM Auckland data began with common problems that hindered the accuracy of the geographic data. The three problems I tackled were standardizing the address data, in particular street names and street types, extracting information about suburbs, and adding the missing city 'Auckland'. After auditing and cleaning, the data provided many useful insights about shops, buildings, sports facilities, and other geographic features.

This rich data could be interesting to any Auckland resident who wants to learn more about her city. Furthermore, the data has potential to supplement market research to inform business development decisions. With improvements, the dataset could be informative enough to help business owners strategize about where to start their new business initiatives.

Appendix

For a list of sources, please refer to the Resources Used.txt file in the Project2 folder in Github (relative path in Github is ../NanoDA/DataWranglingMongoDB/Project2/Resources Used.txt)

"I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."