OpenStreetMap Project
Data Wrangling with MongoDB

Map Area: Johannesburg, Gauteng, South Africa

http://en.wikipedia.org/wiki/Johannesburg
https://www.openstreetmap.org/node/261833893#map=11/-26.2051/28.0497
https://s3.amazonaws.com/metro-extracts.mapzen.com/johannesburg_south-africa.osm.bz2
http://docs.mongodb.org/manual/reference/operator/aggregation-group/
http://docs.mongodb.org/manual/reference/operator/aggregation/
http://docs.mongodb.org/manual/reference/method/db.collection.distinct/

1. Problems encountered with the Map and the dataset

When I first ran the parser I got this error:

```
Traceback (most recent call last):
  File "/Users/CPMcIntyre/Documents/UdacityDataAnalyst/DataWrangling-
Mongodb/osmdb/osmparse.py", line 85, in <module>
    k=shape_element(element)
  File "/Users/CPMcIntyre/Documents/UdacityDataAnalyst/DataWrangling-
Mongodb/osmdb/osmparse.py", line 45, in shape_element
    node['address']['street']=i.get('v')
KeyError: 'address'
```

What it the error was saying is that there was no address key. So I found looked at the code and found it was written like this.

```
for i in element.iter('tag'):
    if i.get('k')=='addr:housenumber':
        node['address']={}
        node['address']['housenumber']=i.get('v')
    elif i.get('k')=='addr:postcode':
```

The error that occurred made two things stand out:

1) In my code, the address dictionary structure was only instantiated if there was a house number.
2) In My dataset, there were elements that had incomplete addresses. Some sub level tags only had a house number, or a street name or the just the city. Each address for each node/way in the dataset was not complete. There were inconsistencies in what data was reported in the OSM xml file.

My solution was for 1*-to instantiate the address dictionary by default. Therefore after the shape data method was run, there would be a point in the process with an empty address dictionary in memory. This dictionary would be populated with keys and values consistent with the data model specified.

My solution for 2* was to fill out the address dictionary structure with the available data in the XML file. The caveat being that not all the fields would be filled out as in the full-fledged template model. Some of the address sub documents in the mongo db will be missing certain fields.

A second issue with my Data is that certain representations of the country South Africa and the city Johannesburg are different in the data.

When I ran this query:

```
j.aggregate([{"$match":{'address':{'$exists':1}}},{'$group':{'_id':'$addr
        ess.city','count':{'$sum':1}}},{'$sort':{'count':-1}}])
```

I returned some representations of Johannesburg I didn't like for example

```
Fourways – Johannesburg
johannesburg
```

I standardized these to just Johannesburg.

The other issue was there was the Dutch/Afrikaans version of South Africa **Zuid-Afrika,** and **ZA** (an abbreviated form). I changed these to be South Africa to be consistent throughout the dataset.

Surprisingly the post codes were all uniform 4 digit entries so there were no need for any refinements.

Through spot Checking the data, I noticed that some of the street conventions were not what I would like them, so I am going to change them using a translate method I implemented that split the string up on whitespaces. From there it substitutes the value I would like to use.

Ave→Avenue,

ave→Avenue,

Ave.→Avenue,

ave.→Avenue,

road→Road

2. Data Overview

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

The Johannesburg data set is 128 MB after it is unzipped.
I used the XML version to parse from

#Number of Entries

j=db.staging #staging is the name of the collection
j.find().count()
660179

Number of nodes

> j.find({"type":"node"}).count()
565573

Number of ways

```
> j.find({"type":"way"}).count()
94591
```

Number of unique users

```
> db.char.distinct({"created.user"}).length
336
```

Top 1 contributing user

```
> k.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},{"$sort":{'count':-
1}},{'$limit':1}])
{ "_id" : "Firefishy", "count" : 65180 }
```

I wanted to get bounds on the latitude and longitude of the city, so I changed the data model in order to get a way to evaluate these changes. The script is attached as 'osmparse-modlatlon.py'

#Max Latitude value

Query:

```
>k.aggregate([{'$match':{'pos.lat':{$exists:1}}},{'$sort':{'pos.lat':-
1}},{'$limit':1},{'$project':{'pos.lat':1}}])
```

Results:

```
{ "_id" : ObjectId("556b5d33331bac5b56aba346"), "pos" : { "lat" : -25.834004 }
```

#Min Latitude value

Query

```
>k.aggregate([{'$match':{'pos.lat':{$exists:1}}},{'$sort':{'pos.lat':1}},{'$limit':1},{'$project':{'pos.lat':1}}])
```

Results:

```
{ "_id" : ObjectId("556b5d70331bac5b56ae0636"), "pos" : { "lat" : -26.5629941 } }
```

#Max Longitude value

Query

```
>k.aggregate([{'$match':{'pos.lon':{$exists:1}}},{'$sort':{'pos.lon':1}},{'$limit':1},{'$project':{'pos.lon':1})
```

Results

```
{ "_id" : ObjectId("556b5d2f331bac5b56ab7795"), "pos" : { "lon" : 28.5419938 }
```

#Min Longitude value

Query

```
>k.aggregate([{'$match':{'pos.lon':{$exists:1}}},{'$sort':{'pos.lon':1}},{'$limit':1},{'$project':{'pos.lon':1})
```

Results
{ "_id" : ObjectId("556b5cf3331bac5b56a915da"), "pos" : { "lon" : 27.5370303 } }

My thoughts on the results of lon/lat comparisons

The results are good because there are no really large outliers. The latitude and longitude are very close together. Referring to the Wikipedia Page-the latitude and longitude are around what is listed.

3. Additional Ideas

Investigation into Contributor area

I am going to compare the boundaries of longitude and latitude that each contributor works in. When we have those boundaries I am going see if there is overlap between the users, or if each user stays in one local area.

Query:
k.aggregate([{'$match':{'pos.lat':{'$exists':1},'pos.lon':{'$exists':1}}},{'$group':
{'_id':'$created.user','maxlat':{'$max':'$pos.lat'},'minlat':{'$min':'$pos.lat'},'maxlon':{'$max':'$pos.lon'},'min
lon':{'$min':'$pos.lon'}}}, {'$sort':{'maxlat':-1}}])

Results
{ "_id" : "NicRoets", "maxlat" : -25.834004, "minlat" : -26.448411, "maxlon" : 28.5417277, "minlon" :
27.538719 }
{ "_id" : "ZoranP", "maxlat" : -25.8340058, "minlat" : -25.8561188, "maxlon" : 28.3068241, "minlon" :
28.2468691 }
{ "_id" : "Adrian Frith", "maxlat" : -25.834006, "minlat" : -26.562992, "maxlon" : 28.54182, "minlon" :
27.5370884 }
{ "_id" : "bahnpirat", "maxlat" : -25.8340112, "minlat" : -26.5629837, "maxlon" : 28.5419938, "minlon" :
27.5371888 }
{ "_id" : "unbeZAhlbar", "maxlat" : -25.8340187, "minlat" : -26.5610141, "maxlon" : 28.5186057,
"minlon" : 27.6189677 }
{ "_id" : "Portur", "maxlat" : -25.834026, "minlat" : -25.8346422, "maxlon" : 28.2460965, "minlon" :
28.2428922 }
{ "_id" : "thebigfatgeek", "maxlat" : -25.8340306, "minlat" : -26.4510496, "maxlon" : 28.5399791,
"minlon" : 27.6117594 }
{ "_id" : "johng", "maxlat" : -25.8340358, "minlat" : -26.5582081, "maxlon" : 28.4243633, "minlon" :
27.5394809 }
{ "_id" : "znh", "maxlat" : -25.8340371, "minlat" : -26.4920112, "maxlon" : 28.4794184, "minlon" :
27.6126305 }
{ "_id" : "user_634020", "maxlat" : -25.8340535, "minlat" : -26.0552037, "maxlon" : 28.2000853,
"minlon" : 28.0216501 }
{ "_id" : "eltonp", "maxlat" : -25.834067, "minlat" : -26.5627858, "maxlon" : 28.500402, "minlon" :
27.5376555 }
{ "_id" : "JacoS", "maxlat" : -25.8340708, "minlat" : -25.9115094, "maxlon" : 28.3182799, "minlon" :
28.1792368 }
{ "_id" : "Deon du Preez", "maxlat" : -25.8340861, "minlat" : -25.8374084, "maxlon" : 27.9309114,
"minlon" : 27.9147524 }
{ "_id" : "LexMid", "maxlat" : -25.8341401, "minlat" : -25.8428074, "maxlon" : 28.3107421, "minlon" :
28.2949101 }
{ "_id" : "Teddy73", "maxlat" : -25.8341467, "minlat" : -26.5623215, "maxlon" : 28.396681, "minlon" :
27.7810016 }

{ "_id" : "Firefishy", "maxlat" : -25.8341571, "minlat" : -26.5629903, "maxlon" : 28.541787, "minlon" : 27.5370303 }
{ "_id" : "MatzeM", "maxlat" : -25.8341911, "minlat" : -26.5625714, "maxlon" : 28.541395, "minlon" : 27.5377111 }
{ "_id" : "Rory Mapstone", "maxlat" : -25.8342444, "minlat" : -26.0870313, "maxlon" : 28.5419596, "minlon" : 27.8076088 }
{ "_id" : "Tinshack", "maxlat" : -25.8342781, "minlat" : -26.5626289, "maxlon" : 28.5402398, "minlon" : 27.591303 }
{ "_id" : "Yolandie(EWCOP)", "maxlat" : -25.8343038, "minlat" : -26.3246144, "maxlon" : 28.2961893, "minlon" : 27.8857354 }
{ "_id" : "dannykath", "maxlat" : -25.8343373, "minlat" : -26.4889736, "maxlon" : 28.3904716, "minlon" : 27.7604395 }
{ "_id" : "Rps333", "maxlat" : -25.834375, "minlat" : -26.1149312, "maxlon" : 28.368684, "minlon" : 28.0357423 }
{ "_id" : "Felefuchs", "maxlat" : -25.8344338, "minlat" : -26.3562657, "maxlon" : 28.3788967, "minlon" : 27.7453919 }
{ "_id" : "marceldup", "maxlat" : -25.8344821, "minlat" : -26.0672015, "maxlon" : 28.353093, "minlon" : 28.0809958 }
{ "_id" : "Gerhardus Geldenhuis", "maxlat" : -25.8344823, "minlat" : -26.5629941, "maxlon" : 28.5419871, "minlon" : 27.539512 }
{ "_id" : "Obelixx", "maxlat" : -25.8345076, "minlat" : -25.9571598, "maxlon" : 28.5416266, "minlon" : 28.2782549 }
{ "_id" : "visage", "maxlat" : -25.8345367, "minlat" : -25.8646897, "maxlon" : 28.3219755, "minlon" : 28.2963939 }
{ "_id" : "Coenie Richards", "maxlat" : -25.8346329, "minlat" : -25.8445619, "maxlon" : 28.3105417, "minlon" : 28.2805304 }
{ "_id" : "gontadu", "maxlat" : -25.8347797, "minlat" : -26.206978, "maxlon" : 28.3658998, "minlon" : 27.9725422 }
{ "_id" : "Basstoelpel", "maxlat" : -25.834785, "minlat" : -26.5613968, "maxlon" : 28.5353398, "minlon" : 27.6645948 }
{ "_id" : "Stheeman", "maxlat" : -25.8348413, "minlat" : -26.5357448, "maxlon" : 28.4310737, "minlon" : 27.5379154 }
{ "_id" : "juergenb22", "maxlat" : -25.8354936, "minlat" : -26.2617599, "maxlon" : 28.2654447, "minlon" : 27.6597869 }
{ "_id" : "Quin", "maxlat" : -25.8356825, "minlat" : -26.4223, "maxlon" : 28.5363835, "minlon" : 27.5432601 }
{ "_id" : "Johan Lubbe", "maxlat" : -25.8357597, "minlat" : -25.8464569, "maxlon" : 28.2756798, "minlon" : 28.252388 }
{ "_id" : "IknowJoseph", "maxlat" : -25.835841, "minlat" : -26.2685821, "maxlon" : 28.1956766, "minlon" : 27.9809791 }
{ "_id" : "Andreas Pauley", "maxlat" : -25.8358937, "minlat" : -25.8455721, "maxlon" : 28.2722406, "minlon" : 28.1945045 }
{ "_id" : "Duplicate Account - Coenie Richards (New)", "maxlat" : -25.8359327, "minlat" : -25.8463713, "maxlon" : 28.3225965, "minlon" : 28.3050283 }
{ "_id" : "SuneMomsen", "maxlat" : -25.8360315, "minlat" : -26.5013988, "maxlon" : 28.466775, "minlon" : 27.6513452 }
{ "_id" : "bvivi", "maxlat" : -25.8360345, "minlat" : -26.5046736, "maxlon" : 28.2790988, "minlon" : 27.837865 }
{ "_id" : "BennieD", "maxlat" : -25.8361107, "minlat" : -26.5374861, "maxlon" : 28.5382954, "minlon" : 27.7697958 }

Analysis:
Just skimming over the entries for each user and keeping in mind the global max and mins of both longitude and latitude, it doesn't appear that each user was confined to a small area. The locations tendered by the users suggest that each of the users entered data points from all over Johannesburg.

Investigation into Contributor preferences

I am going to look into each contributor to see if any specific contributor looks are a certain amenity more than another. For example, maybe one of the contributor really like restaurants, so they would add a large quantity of restaurants to the data. Maybe another contributor really enjoys public parks and recreation areas-therefore they would contribute to a lot of that type of amenity.

Query:

```
> k.aggregate([{'$match':{'amenity':{'$exists':1}}},{'$group':{'_id':{'user':'$created.user','amenity
type':'$amenity'},'count':{'$sum':1}}},{'$sort':{'user':-1}},{'$sort':{'count':-1}}])
```

Results:

```
{ "_id" : { "user" : "SuneMomsen", "amenity type" : "parking" }, "count" : 230 }
{ "_id" : { "user" : "CharlizeFerreira", "amenity type" : "parking" }, "count" : 225 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "restaurant" }, "count" : 118 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "fast_food" }, "count" : 95 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "fuel" }, "count" : 76 }
{ "_id" : { "user" : "thomasF", "amenity type" : "swimming_pool" }, "count" : 75 }
{ "_id" : { "user" : "Firefishy", "amenity type" : "fuel" }, "count" : 74 }
{ "_id" : { "user" : "JacquesFauré", "amenity type" : "parking" }, "count" : 73 }
{ "_id" : { "user" : "Gerhardus Geldenhuis", "amenity type" : "parking" }, "count" : 63 }
{ "_id" : { "user" : "Tinshack", "amenity type" : "parking" }, "count" : 58 }
{ "_id" : { "user" : "Dirk Momsen", "amenity type" : "parking" }, "count" : 55 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "bank" }, "count" : 52 }
{ "_id" : { "user" : "CharlizeFerreira", "amenity type" : "restaurant" }, "count" : 48 }
{ "_id" : { "user" : "DawidLoubser", "amenity type" : "parking" }, "count" : 48 }
{ "_id" : { "user" : "CharlizeFerreira", "amenity type" : "fast_food" }, "count" : 45 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "place_of_worship" }, "count" : 44 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "atm" }, "count" : 43 }
{ "_id" : { "user" : "Rgeers", "amenity type" : "parking" }, "count" : 42 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "cafe" }, "count" : 41 }
{ "_id" : { "user" : "SuneMomsen", "amenity type" : "fast_food" }, "count" : 38 }
{ "_id" : { "user" : "SuneMomsen", "amenity type" : "restaurant" }, "count" : 37 }
{ "_id" : { "user" : "Firefishy", "amenity type" : "school" }, "count" : 37 }
{ "_id" : { "user" : "DawidLoubser", "amenity type" : "restaurant" }, "count" : 36 }
{ "_id" : { "user" : "dogmatic69", "amenity type" : "parking" }, "count" : 36 }
{ "_id" : { "user" : "Firefishy", "amenity type" : "parking" }, "count" : 36 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "pharmacy" }, "count" : 34 }
{ "_id" : { "user" : "slashme", "amenity type" : "parking" }, "count" : 30 }
{ "_id" : { "user" : "Tinshack", "amenity type" : "school" }, "count" : 27 }
{ "_id" : { "user" : "titanbeos", "amenity type" : "school" }, "count" : 27 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "school" }, "count" : 26 }
{ "_id" : { "user" : "GJS-za", "amenity type" : "fuel" }, "count" : 26 }
{ "_id" : { "user" : "slashme", "amenity type" : "restaurant" }, "count" : 25 }
{ "_id" : { "user" : "William_", "amenity type" : "parking" }, "count" : 24 }
{ "_id" : { "user" : "Walter Schlögl", "amenity type" : "parking" }, "count" : 23 }
{ "_id" : { "user" : "Walter Schlögl", "amenity type" : "restaurant" }, "count" : 22 }
```

{ "_id" : { "user" : "SuneMomsen", "amenity type" : "atm" }, "count" : 20 }
{ "_id" : { "user" : "Gary Alexander", "amenity type" : "fuel" }, "count" : 20 }
{ "_id" : { "user" : "Stephen Reynolds", "amenity type" : "parking" }, "count" : 20 }
{ "_id" : { "user" : "SuneMomsen", "amenity type" : "pharmacy" }, "count" : 20 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "post_office" }, "count" : 19 }
{ "_id" : { "user" : "slashme", "amenity type" : "atm" }, "count" : 19 }
{ "_id" : { "user" : "JacquesFauré", "amenity type" : "fast_food" }, "count" : 18 }
{ "_id" : { "user" : "Tinshack", "amenity type" : "place_of_worship" }, "count" : 18 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "pub" }, "count" : 17 }
{ "_id" : { "user" : "Tomas Straupis", "amenity type" : "fuel" }, "count" : 17 }
{ "_id" : { "user" : "Mark Spark", "amenity type" : "school" }, "count" : 17 }
{ "_id" : { "user" : "johng", "amenity type" : "parking" }, "count" : 16 }
{ "_id" : { "user" : "bri g", "amenity type" : "fuel" }, "count" : 16 }
{ "_id" : { "user" : "Tinshack", "amenity type" : "fuel" }, "count" : 16 }
{ "_id" : { "user" : "dj015", "amenity type" : "fuel" }, "count" : 16 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "parking" }, "count" : 16 }
{ "_id" : { "user" : "BennieD", "amenity type" : "school" }, "count" : 15 }
{ "_id" : { "user" : "slashme", "amenity type" : "school" }, "count" : 15 }
{ "_id" : { "user" : "slashme", "amenity type" : "fast_food" }, "count" : 15 }
{ "_id" : { "user" : "unbeZAhlbar", "amenity type" : "school" }, "count" : 14 }
{ "_id" : { "user" : "NicRoets", "amenity type" : "kindergarten" }, "count" : 14 }
{ "_id" : { "user" : "SuneMomsen", "amenity type" : "bank" }, "count" : 14 }
{ "_id" : { "user" : "LordTrilobite", "amenity type" : "parking" }, "count" : 13 }
{ "_id" : { "user" : "DawidLoubser", "amenity type" : "fuel" }, "count" : 13 }
{ "_id" : { "user" : "user_634020", "amenity type" : "fuel" }, "count" : 13 }

Analysis:
There did not appear to be a specific user that had preference for a specific type of amenity. That means that it doesn't appear that a restaurant enthusiast is only logging data points for restaurants. Rather, the behavior is that the users who log more data points occur more frequently among all the different amenity types.

Conclusion

After reviewing this data, it is clear that there is a lot of data missing for addresses specifically. Looking at other data point attributes such as latitude and longitude, it appears that the values are appropriate and within the generally acceptable rage for Johannesburg (based on Wikipedia latitude and longitudinal data). Similarly the postal codes listed in the address ranges are uniform and belong to Johannesburg. The cleaning process has helped get the data to a stage that is helpful, but it is not robust enough to use for something like an app.