Final Project: Data Wrangling with MongoDB

Problems Encountered in My Map

1. Some of the cities are misspelled. For example, the data include “Little Elm” which is correct and “Little ELm”. These names were cleaned when generating the json from xml. See #update_city_name in data.py.
2. Another example is that a few of the city designations include the state. For example, “Plano, TX” instead of just “Plano”. I cleaned the city names of this when generating the json from xml. Also see #update_city_name in data.py.
3. 251 items have a website listed. But the website format is somewhat inconsistent. I corrected that by making sure the website is listed as a valid url for all web properties. See #update_website in data.py.

Overview of the Data

File size:
downloaded xml: 87 MB
json representation: 95 MB

Unique Users:

```
> db.osm.distinct('created.user').length
504
```

As in SQL, MongoDB’s distinct is obviously useful for this.

Number of Nodes:

```
> db.osm.find({'type':'node'}).count()
366725
```

Number of Ways:

```
> db.osm.find({'type':'way'}).count()
41243
```

Number of distinct cities:

```
> db.osm.distinct('address.city').length
17
> db.osm.distinct('address.city')
[
      "Plano",
      "Frisco",
      "Richardson",
      "Carrollton",
      "Sachse",
```

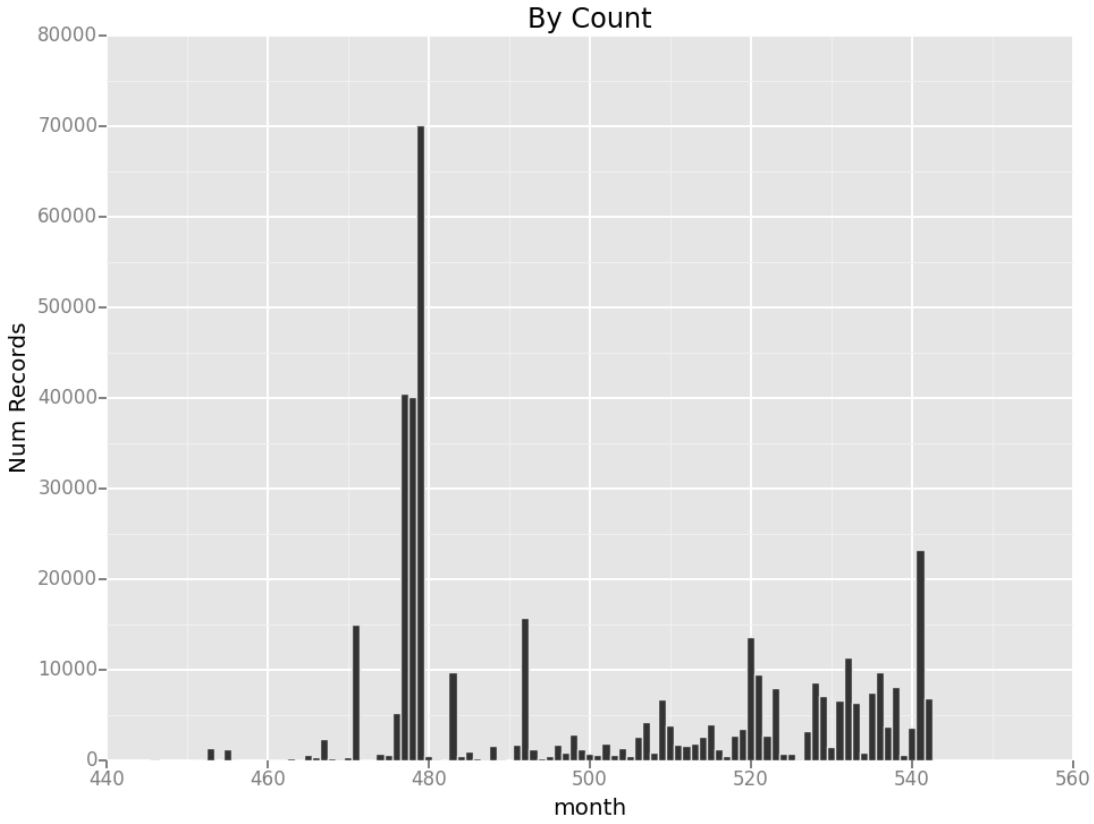
```
        "Addison",
        "Dallas",
        "McKinney",
        "Murphy",
        "Garland",
        "Wylie",
        "Allen",
        "Lewisville",
        "Coppell",
        "The Colony",
        "Little Elm",
        "Mckinney"
]
```

When was this data created?:

The data range from March 2007 to March 2015 (this year).

```
> db.osm.find().sort({'created.timestamp':1}).limit(1)
{ "_id" : ObjectId("550dfbcc445faa134c2142e4"), "created" : { "changeset" : "232647", "user" :
"user_6514", "version" : "1", "uid" : "6514", "timestamp" : "2007-03-09T23:15:37Z" }, "type" :
"node", "id" : "26450262", "pos" : [ 32.9905615, -97.0033364 ] }
>
> db.osm.find().sort({'created.timestamp':-1}).limit(1)
{ "_id" : ObjectId("550dfbcd445faa134c21b57c"), "created" : { "changeset" : "29630436",
"user" : "eugenebata", "version" : "3", "uid" : "624003", "timestamp" :
"2015-03-21T08:37:21Z" }, "type" : "node", "id" : "81817529", "pos" : [ 32.7778195,
-96.986125 ] }
```

I plotted a histogram of activity related to this data over time aggregating by month. (see explore.py) I thought it might be interesting, and might help track down issues to see if there were a pattern to when the data were created or edited. The histogram is below:



The data range in this histogram is from month 446 to month 542. We see a very large spike in activity in months 477 through 479 corresponding to approximately September through November of 2009.
There also seems to be somewhat more activity in the last year or so. This might be an indication of a trend towards increasing use of OpenStreetMap.

Number of street addresses:
Of the 372,267 nodes and closed ways, only 3,906 have addresses

```
> db.osm.find({'address.street':{'$exists':1}}).count()
3906
```

Node types / amenities:
As in the example project, I also looked at the top amenities.
Only 2,029 entries in the dataset have an amenity attribute.

```
> count = db.osm.find({'amenity': {'$exists':1}}).count()
2029
```

Looking at the top amenities ...

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

```
{ "_id" : "parking", "count" : 545 }
{ "_id" : "restaurant", "count" : 289 }
{ "_id" : "school", "count" : 249 }
{ "_id" : "fast_food", "count" : 249 }
{ "_id" : "place_of_worship", "count" : 170 }
{ "_id" : "fuel", "count" : 94 }
{ "_id" : "bank", "count" : 73 }
{ "_id" : "pharmacy", "count" : 37 }
{ "_id" : "cafe", "count" : 33 }
{ "_id" : "post_box", "count" : 30 }
{ "_id" : "post_office", "count" : 28 }
{ "_id" : "fire_station", "count" : 26 }
{ "_id" : "fire_hydrant", "count" : 20 }
{ "_id" : "grave_yard", "count" : 18 }
{ "_id" : "fountain", "count" : 16 }
{ "_id" : "hospital", "count" : 15 }
{ "_id" : "bench", "count" : 14 }
{ "_id" : "library", "count" : 12 }
{ "_id" : "police", "count" : 10 }
{ "_id" : "cinema", "count" : 9 }
```

In my estimation, these counts point to the incompleteness of the data. Obviously, there are more than 20 fire hydrants in the area surveyed. This is just one example. Parking seems over represented to me. I speculated that there was some agenda associated with that, but a quick sampling of those data not show any obvious pattern, e.g. that they were all created by the same user.
I did find that all of the “fire_hydrant” nodes were from the same user and along the same street. And there really are fire hydrants there, I confirmed by plugging a few of them into Google Maps using the lat/lon and examining in “street view” within Google Maps.

Other Ideas About the Dataset

I noticed that there are many ways in this data set that are areas, i.e. they begin and end with the same node. These areas are used to represent things like buildings and parking areas. There is no easy way to query open ways versus closed ways once the data is in MongoDB. I found this would be useful so I added an “open” boolean field and populated that by inspecting the nodes of the way. This was most easily done while transforming the data from xml into json prior to import into the database. See data.py line 226.
And now we can look at closed ways if we are looking for information more related to places: Of the 41,243 ways, 35,701 are open and 5,542 are closed.

```
> db.osm.find({'type':'way', 'open':false}).count()
5542
```

Furthermore I found that postcode could be found on some of the closed ways. In other projects I have worked on (and continue to work on) I see some potential in tying together postcode and lat/lon. For things like buildings, it would be sufficient to simply grab one of the nodes listed in node_refs and use the position of that node (in lat/lon) as the lat/lon for the postcode. You could then take all the nodes for a given postcode and find the center to get a reasonable approximation for the lat/lon for the postcode.
There would be some challenges there. The largest challenge is probably that there aren't enough data points with postcode information. But that may change over time. A second challenge is the technical challenge of finding the center given a set of lat/lon pairs. But we could use scikit.learn DBSCAN, or possibly some other algorithm or approach.