OpenStreetMap Project
Data Wrangling with MongoDB

James Ward

Map Area: Albuquerque, NM, USA

https://www.openstreetmap.org/search?query=Albuquerque%2C%20NM#map=12/35.0845/-106.6511
https://mapzen.com/data/metro-extracts/#albuquerque-new-mexico

Contents

Problems Encountered in the Data

In order to avoid working with representative subsamples of the data, I chose Albuquerque, New Mexico as a relatively small dataset so that I could analyze the entire dataset holistically along the way. Before looking for issues in the data, I first did some basic exploration to get a sense for what the data looks like. Here are the top tags contained in the elements:

```
Tags contained:
       [('nd', 367230),
        ('node', 292869),
        ('tag', 237428),
```

```
 ('way', 44966),
 ('member', 3178),
 ('relation', 644),
 ('bounds', 1),
 ('osm', 1)]
```

And here are the top contributing users:

```
Top 10 contributing users:
[('EdHillsman', 113327),
 ('anjbe', 68935),
 ('jackbus', 30624),
 ('woodpeck_fixbot', 27784),
 ('beweta', 12422),
 ('greenv505', 10223),
 ('polarsleuth', 9296),
 ('Dilys', 4126),
 ('balrog-kun', 3884),
 ('derricknehrenberg', 3071)]
```

Upon iterating through the elements, here are a few examples of how the elements and tags are structured:

```
<Element 'node' at 0x00000000B4606AE0>
{'changeset': '2659832', 'uid': '97431', 'timestamp': '2009-09-28T01:42:58Z',
'lon': '-106.7143614', 'version': '3', 'user': 'Dion Dock', 'lat':
'35.1059881', 'id': '140839970'}
<Element 'tag' at 0x00000000FD47A1E0>
{'k': 'source', 'v': 'Yahoo'}
<Element 'tag' at 0x00000000FD47A210>
{'k': 'highway', 'v': 'turning_circle'}

<Element 'node' at 0x0000000059B70690>
{'changeset': '27584075', 'uid': '502086', 'timestamp': '2014-12-20T08:54:50Z',
'lon': '-106.5137876', 'version': '3', 'user': 'anjbe', 'lat': '35.0629584',
'id': '140902747'}
<Element 'tag' at 0x0000000059B706F0>
{'k': 'traffic_calming', 'v': 'hump'}

<Element 'way' at 0x000000009E688EA0>
{'changeset': '6539697', 'uid': '736', 'timestamp': '2010-12-04T16:14:28Z',
'version': '10', 'user': 'Steve Chilton', 'id': '1641009'}
<Element 'nd' at 0x000000009E688F00>
{'ref': '13878341'}
<Element 'nd' at 0x000000009E688F60>
{'ref': '13878340'}
<Element 'nd' at 0x000000009E688FC0>
{'ref': '13878338'}
<Element 'tag' at 0x000000009E68B030>
{'k': 'highway', 'v': 'footway'}

<Element 'way' at 0x00000000EAB62EA0>
{'changeset': '4313608', 'uid': '20587', 'timestamp': '2010-04-03T14:33:11Z',
'version': '2', 'user': 'balrog-kun', 'id': '14403157'}
<Element 'nd' at 0x00000000EAB62F00>
{'ref': '140740448'}
<Element 'nd' at 0x00000000EAB62F60>
```

```
{'ref': '140740451'}
<Element 'nd' at 0x00000000EAB62FC0>
{'ref': '140740452'}
<Element 'nd' at 0x00000000EAB67060>
{'ref': '140740453'}
<Element 'tag' at 0x00000000EAB67090>
{'k': 'name', 'v': 'Johannes Kepler Drive'}
<Element 'tag' at 0x00000000EAB670F0>
{'k': 'highway', 'v': 'residential'}
<Element 'tag' at 0x00000000EAB67150>
{'k': 'tiger:cfcc', 'v': 'A41'}
<Element 'tag' at 0x00000000EAB671B0>
{'k': 'tiger:tlid', 'v': '206784874:206792855:206792856'}
<Element 'tag' at 0x00000000EAB67240>
{'k': 'tiger:county', 'v': 'Bernalillo, NM'}
<Element 'tag' at 0x00000000EAB672A0>
{'k': 'tiger:source', 'v': 'tiger_import_dch_v0.6_20070828'}
<Element 'tag' at 0x00000000EAB67300>
{'k': 'tiger:reviewed', 'v': 'no'}
<Element 'tag' at 0x00000000EAB67390>
{'k': 'tiger:name_base', 'v': 'Johannes Kepler'}
<Element 'tag' at 0x00000000EAB673F0>
{'k': 'tiger:name_type', 'v': 'Dr'}
<Element 'tag' at 0x00000000EAB67450>
{'k': 'tiger:separated', 'v': 'no'}
```

The primary elements of the map data appear to be the “node” and “way” elements, with many sub-tags beneath them. Following this structure, I was able to iterate through them to extract and examine the common key:value pairs for the “tag” sub-tags. Here are some representative examples:

```
Node elements:
       name >> Street Food Asia
       phone >> +1-505-260-0088
       amenity >> restaurant
       website >> http://www.streetfoodasia.com/
       addr:street >> Central Avenue Southeast
       addr:postcode >> 87106
       opening_hours >> Su-Th 11:00-22:00; Fr-Sa 11:00-23:00
       addr:housenumber >> 3422

       name >> Piatanzi
       note >> Name changed to Piatanzi from Piattini. See
       http://www.abqjournal.com/599130/abqnewsseeker/forget-piattini-its-now-
       piatanzi.html for details.
       phone >> 505-792-1700
       amenity >> restaurant
       cuisine >> Italian
       website >> http://www.piatanzi.com/
       addr:city >> Albuquerque
       addr:state >> NM
       addr:street >> Girard Boulevard Northeast
       addr:postcode >> 87106
       addr:housenumber >> 1403

       power >> tower
       source >> Yahoo

       addr:street >> Flora Street Southwest
       addr:postcode >> 87121
       addr:housenumber >> 3139
```

```
        highway >> crossing

        bicycle >> yes
        highway >> crossing
        crossing >> traffic_signals

Way elements:
        highway >> footway

        name >> J D Sports (Shop)
        building >> store
        created_by >> JOSM

        name >> Waverley Road
        lanes >> 2
        highway >> residential
        surface >> asphalt

        name >> Menlo Court
        highway >> residential
        created_by >> JOSM
        tiger:cfcc >> A41
        tiger:tlid >> 155514076
        tiger:source >> tiger_import_dch_v0.6_20070808
        tiger:reviewed >> no
        tiger:zip_left >> 97504
        tiger:name_base >> Menlo
        tiger:name_type >> Ct
        tiger:separated >> no
        tiger:zip_right >> 97504
```

These show the flexibility of the JSON format. Each element only contains the tags where data has been added – there are no mandatory tags, or empty tags where no data exists for a particular field. Upon scanning through the entire dataset, the "addr:street" tag under the node element appears to be particularly important from a data quality standpoint, and is a primary focus of the analysis from here on out.

After doing some further data analysis in the python notebook, I noticed the following unexpected issues:

1. Extensive use of directional indicators (Northwest, SE, etc.)
2. Addresses that contain prepositions and descriptive phrases ('Gold Between 1st and 2nd', 'Corner of Uptown Blvd and Uptown Loop', etc.)
3. Addresses that use an area or landmark for the street name ('UNM Hospitals, Albuquerque', 'San Felipe', 'Indian School'
4. Punctuation, spelling or capitalization mistakes ('Central avenue', 'Comanche Road Norhteast', 'Cental AvenueSW', etc.)
5. Improper use of the "Street" field ('11115', '4th Street NW Suite 250', '1833 8th Street NorthwestAlbuquerque, NM 87102', etc.)

UNEXPECTED "STREET TYPES", AFTER REMOVING STANDARD DIRECTIONAL INDICATORS (E.G. "NORTHEAST"):

```
{'11115': set(['11115']),
 '250': set(['4th Street NW Suite 250']),
 '2nd': set(['Gold Between 1st and 2nd']),
 '5th': set(['5th']),
```

```
'87102': set(['1833 8th Street NorthwestAlbuquerque, NM 87102']),
'A': set(['3301 Menaul Blvd. NE Suite A', 'Juan Tabo NE, Suite A']),
'Albuquerque': set(['UNM Hospitals, Albuquerque']),
'Ave': set(['8700 Central Ave']),
'AvenueSW': set(['Cental AvenueSW']),
'Basehart': set(['1401 Basehart']),
'Central': set(['8th and Central']),
'E-18': set(['Eubank Northeast Ste E-18']),
'East': set(['Redondo East']),
'Felipe': set(['San Felipe']),
'Freeway': set(['Pan American Freeway']),
'Johann-Gutenberg-Gasse': set(['Johann-Gutenberg-Gasse']),
'Loop': set(['Corner of Uptown Blvd and Uptown Loop']),
'Morningside': set(['Lead and Morningside']),
'NE': set(['1915 Roma Ave. NE',
          …… (14 more instances)
           'Uptown Loop Rd NE']),
'NE.': set(['Menaul Blvd. NE.']),
'NW': set(['Valley View Dr NW']),
'Neubau': set(['Neubau']),
'Norhteast': set(['Comanche Road Norhteast']),
'Oeste': set(['Vista Oeste']),
'Pl': set(['Mullhacen Pl']),
'SE': set(['3400 Crest Ave. SE',
           ……  (4 more instances)
           'Randolph Ave SE']),
'SW': set(['Bridge Boulevard SW']),
'School': set(['Indian School']),
'SouthWest': set(['16th Street SouthWest']),
'Vista': set(['Buena Vista']),
'Yale': set(['Yale']),
'avenue': set(['Central avenue'])}
```

Extensive use of directional indicators

This is not really a problem, so much as an unexpected finding in the data. I modified my "shape_element" function to standardize all directional indicators into the format of "Northeast", "Southwest", etc.

Addresses that contain prepositions and descriptive phrases

In the absence of better information, I left these alone – the most effective way to address these would be to have a person review them on a case-by-case basis to determine if there is a more appropriate way to specify the address. Some of the descriptions seem to suggest that the location may not have an official postal address. On the other hand, some seem to contain information that will be useful in determining the true value. For example, 'Gold Between 1st and 2nd' might become 'Gold Street' or 'Gold Avenue'.

Addresses that use an area or landmark as the street name

Again, without better information to go on, I left these alone. It is easy to create an algorithm to find these anomalies, but it usually takes some human analysis to determine if the value is actually wrong,

and if so, what it should actually be. It would be helpful for any human interfaces to this data (e.g. Google Maps) to call out these suspected errors so people can easily see and correct them.

Punctuation, spelling and capitalization mistakes

Some of these can be discovered and corrected programmatically, but many are more subtle and difficult to find without knowing exactly what you are looking for. My methodology detected some of these with regular expression searches, and corrected some with the update_name function, but this was certainly not comprehensive.

Improper use of the street field

There are various errors here, such as inclusion of a suite or apartment number, using the street field for an entirely different piece of data (zip code), or putting the entire address into the street field. It is difficult to come up with a full-proof automated method to detect and correct these mistakes, since they are so varied in nature. Again, the best solution is probably to have an algorithm to call out these suspected mistakes so that a person can review and address them.

Data Overview

The following are some descriptive characteristics of the data in question:

albuquerque.osm	OSM File	68,343 KB
albuquerque.osm.json	JSON File	76,575 KB

NUMBER OF DOCUMENTS
```
> db.albu.find().count()
337835
```

NUMBER OF NODES
```
> db.albu.find({"type":"node"}).count()
292866
```

NUMBER OF WAYS
```
> db.albu.find({"type":"way"}).count()
44961
```

NUMBER OF UNIQUE USERS
```
> db.albu.distinct("created.user").length
604
```

TOP 10 CONTRIBUTING USERS
```
> db.albu.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":10}])
{ "_id" : "EdHillsman", "count" : 113309 }
{ "_id" : "anjbe", "count" : 68856 }
{ "_id" : "jackbus", "count" : 30624 }
```

```
{ "_id" : "woodpeck_fixbot", "count" : 27784 }
{ "_id" : "beweta", "count" : 12420 }
{ "_id" : "greenv505", "count" : 10223 }
{ "_id" : "polarsleuth", "count" : 9295 }
{ "_id" : "Dilys", "count" : 4119 }
{ "_id" : "balrog-kun", "count" : 3884 }
{ "_id" : "derricknehrenberg", "count" : 3071 }
```

NUMBER OF USERS APPEARING ONLY ONCE (HAVING 1 POST)

```
> db.albu.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}},
{"$limit":1}])
{ "_id" : 1, "num_users" : 178 }
# "_id" represents postcount
```

TOP POSTAL CODES

```
> db.albu.aggregate([{"$match":{"address.postcode":{"$exists":1}}}, {"$group":{"
_id":"$address.postcode", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])
{ "_id" : "87123", "count" : 599 }
{ "_id" : "87110", "count" : 377 }
{ "_id" : "87121", "count" : 362 }
{ "_id" : "87106", "count" : 270 }
{ "_id" : "87108", "count" : 218 }
…
```

All values appear reasonable (e.g. nearly all postal codes are valid, contributions are spread over 604 users with the "lion's share" going to the top few, etc.). I also noticed that many tags have keys that start with "tiger:", such as "'tiger:county" and "tiger:name_base". By doing some basic internet searches, I found that these tags come from large imports from the Topologically Integrated Geographic Encoding and Referencing system (TIGER) data, produced by the US Census Bureau. It looks like these imports occurred multiple times, and actually had to be rolled back at one point early in the process due to data quality issues.

Additional Ideas

Ease of Automatically Detecting Errors vs. Automatically Fixing Errors

In reviewing the data, it was relatively easy to surface data quality issues through automated methods by using and refining regular expression searches. However, automatically fixing the errors turned out to be much more problematic because the correct information was often not present. In the case of abbreviations, format variations or common spelling errors, I was able to easily substitute the better values programmatically in a short amount of time.

However, many of the errors cannot be easily corrected without much more advanced logic, or a person evaluating each of them on a case-by-case basis. For instance, there are 3 instances of the postal code "2000", which is obviously incorrect – however, the correct value is not contained in any of the other tags, so we would have to write a function to cross-check this against another data source, or have a person search each address and put in the correct value. Some street values are also problematic: 'Corner of Uptown Blvd and Uptown Loop' (which street is the address actually on?), 'Johann-Gutenberg-Gasse' (this takes translation to determine if it is a valid street name – 'Gasse' roughly

translates to ‘Lane’ in German) and 'Neubau' (is this a street name? maybe ‘Neubau Street’? or maybe a neighborhood?) are some examples.

Calling Out Errors in User Interface

Related to the observation above, one approach to addressing these more problematic errors would be to prompt people for corrections whenever they view map data that has suspected errors. This could be done in whatever user interface they are using to interact with the data, but must obviously be designed as to not be too obtrusive. This would effect a mixed data quality approach where algorithms are used for most error detection and some error correction, while human intelligence would serve as the “back-stop” for occasional tougher errors that cannot be resolved programmatically. This could also be combined with some gamification aspects to encourage users to compete in improving the data (similar to Waze and many other modern applications).

Conclusion

The Albuquerque Openstreetmap dataset appears to be fairly healthy, with a relatively small number of data quality issues. However, in reviewing the data, I also noticed that there are a large number of elements that have very few tags associated with them. For instance, most stores appear to have only very basic information (lat-lon position, name, and maybe one or two other tags), while some stores have much more thorough information (street address, website, store hours, etc.). This highlights both the flexibility and the inconsistency of the JSON data format. Hopefully over time, with more use, and potentially with some of the user interface prompts and gamification concepts mentioned above, users will build this dataset to be a more complete and useful representation of the Albuquerque area.