Data Analyst Nano Degree Project 2
December Cohort 2014
Fang
March 30, 2015

Map Area: Austin, TX USA
https://mapzen.com/metro-extracts/

1. Problems Encountered in the Map Data

After modifying the audit code from the lesson 6 case study to more easily audit the types of tags and their contents I chose to investigate the same categories as in the sample exercise. Similar issues appeared as follows:

- Over-abbreviated Street names
- Inconsistent representation of postal codes/zip codes
- Inconsistent presentation of state and city names
- Uniformity of types of amenity and cuisine

Over-abbreviated Street Names:

Modification to the original code was made to process all the substrings of street names to remove over-abbreviation. Also normalized the way certain highways are represented.

Example:
N Greystone Dr Ste 210 -> North Greystone Drive Suite 210

Inconsistent Representation of Postal Codes/Zip Codes:

As the sample project found, zip code representation is often inconsistent containing the state abbreviation and the plus four codes. To promote consistency, all postal codes were stripped to their 5 digit baseline.

Interesting postcode “14150” appeared in the postcode audit which is associated with Tonawanda, NY. The postcode was left during first processing because it was a legitimate 5-digit code. After looking into the node entry in Mongodb yield a car shop with the website “www.nylemaxwellcjd.com”. A subsequent google search yielded a Taylor (adjacent to Austin) address of **14150** State Highway 79, Taylor, TX 76574. The conclusion is that there was a mis-entry of the street number, and a special case was added to change the zip to 76574.

Inconsistent Representation of State and City Names:

State name capitalization and abbreviation inconsistencies were normalized for the state name entry. Entries of “Tx”, “tX”, “Texas” were all transformed to the standard state abbreviation “TX”.

City names were contaminated with state information and even one that included USA: “Austin;TX;USA”. Some city names were misspelled such as Westlake Hills where the correct spelling is West Lake Hills where there is separation between West and Lake. All city names were purged of state and country information and misspellings fixed.

Uniformity of types of amenity and cuisine:

Amenity and cuisine are two interesting tags that I wanted cleaned for purpose of interest. Minor issues were found with capitalization. All cuisine and amenity names were normalized to be lower case and have “_” in place of spaces. I did take the liberty to consolidate “bbq”, “Bar-b-q” and “barbecue” into “barbecue” category under cuisine.

An interesting amenity caught my eye during the audit: ‘yes’. Leaving the amenity type intact, i found the entry in mongodb:

```
db.austin_texas.find({"amenity": "yes"}).pretty()
{
        "_id" : ObjectId("5518f9b6145603dc15bc8a3a"),
        "amenity" : "yes",
        "name" : "European Wax Center",
        "created" : {
                "uid" : "703517",
                "changeset" : "25571519",
                "version" : "2",
                "user" : "Iowa Kid",
                "timestamp" : "2014-09-21T04:33:06Z"
        },
        "pos" : [
                30.2008883,
                -97.8760776
        ],
        "visible" : "false",
        "type" : "node",
        "id" : "2793696842"
}
```

While I cannot argue that this is certainly an amenity, I found it very hard to categorize. Looking through the list of amenity types in the Austin area there wasn't a good fit for a European Wax Center. I finally settled on "health_and_well-being" or just "other", and adjusted the cleaning code to process this one instance, but decided to comment out the line because it didn't seem quite right to hardcode all "yes" amenities (even though there was only one) to one particular amenity type. I felt this particular fix should be performed after the processing, case by case.

2. Data Overview

This section contains statistics on the data and the mongodb queries used to obtain them.

File Size:

austin_texas.osm …….. 170mb

I chose to ingest to mongodb using insert in consideration of future usability of the code. Below is the statistics of the resulting collection "austin_texas" with size on disk of 251mb.

```
db.austin_texas.stats()
{
        "ns" : "mydb.austin_texas",
        "count" : 859061,
        "size" : 263163312,
        "avgObjSize" : 306,
        "storageSize" : 335900672,
        "numExtents" : 14,
        "nindexes" : 1,
        "lastExtentSize" : 92585984,
        "paddingFactor" : 1,
        "systemFlags" : 1,
        "userFlags" : 1,
        "totalIndexSize" : 27888336,
        "indexSizes" : {
                "_id_" : 27888336
        },
        "ok" : 1
}
```

#Number of Documents

```
> db.austin_texas.find().count()
859061
```

#Number of Nodes

```
> db.austin_texas.find({"type":"node"}).count()
778680
```

#Number of Ways

```
> db.austin_texas.find({"type":"way"}).count()
80371
```

#Number of Unique Users

```
> db.austin_texas.distinct("created.user").length
884
```

#Top 3 Contributing Users

```
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}}, {"$sort": {"count":
-1}},{"$limit": 3}]
result = collection.aggregate(pipeline)
pprint.pprint(result["result"])

[{u'_id': u'woodpeck_fixbot', u'count': 243497},
 {u'_id': u'varmint', u'count': 38856},
 {u'_id': u'richlv', u'count': 37953}]
```

#Number of Users Contributing Only 1 Post

```
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
            {"$group": {'_id': "$count", "num":{"$sum":1}}},
            {"$match":{"_id": 1}},
            {"$limit": 3}]
result = collection.aggregate(pipeline)
pprint.pprint(result)

[{u'_id': 1, u'num': 169}]
```

3. Additional Ideas

User Contribution to the 859061 total documents

Top 3 Combined:
Contributed: 320306
Percentage of Total: 37.2855943874

Top 10 Combined:
Contributed: 504959
Percentage of Total: 58.8%

Top 20 Combined:
Contributed: 628815
Percentage of Total: 73.2%

There seems to be a skewed distribution to the contribution of OpenStreetMap. Personally to drive involvement, if an interactive application on the phone could be provided with easy to contribute options, there would be more user involvement from more people. I am reminded of Waze, where it is a social traffic and gps application that gets a lot of usage and participation.

Additional Data Exploration

#Top 5 Appearing Amenities

```
pipeline = [{"$match": {"amenity":{"$exists":1}}},
            {"$group": {"_id": "$amenity", "count": {"$sum": 1}}},
            {"$sort": {"count": -1}},
            {"$limit": 5}]
result = collection.aggregate(pipeline)
pprint.pprint(result["result"])
```

```
[{u'_id': u'parking', u'count': 1847},
 {u'_id': u'restaurant', u'count': 684},
 {u'_id': u'waste_basket', u'count': 591},
 {u'_id': u'school', u'count': 574},
 {u'_id': u'fast_food', u'count': 498}]
```

The list comes as no large surprise as parking is a big issue in Austin. Food option are well represented by number 2 and 5. Austin is also very green, the “waste_basket” amenity coming in at number 3 was some of a surprise, but also very logical when examined.

#Top 5 type of Cuisine

```
pipeline = [{"$match": {"amenity":{"$exists":1}, "amenity":"restaurant"}},
            {"$group": {"_id": "$cuisine", "count": {"$sum": 1}}},
            {"$sort": {"count": -1}},
            {"$limit": 5}]
result = collection.aggregate(pipeline)
pprint.pprint(result["result"])
```

```
[{u'_id': None, u'count': 373},
 {u'_id': u'mexican', u'count': 68},
 {u'_id': u'american', u'count': 25},
 {u'_id': u'pizza', u'count': 23},
 {u'_id': u'chinese', u'count': 20}]
```

I hesitated to do an extra “$match: $exists” line for cuisine, and the result was to discover that not all restaurants have a cuisine type. I’m mildly surprised that “barbecue” is not higher on the list as it is very popular in Austin with many famous barbecue establishments (it’s #10 with None as the top, in truth #9).

4. Conclusion

Fully cleaning the data will require a systematic audit of each and every tag and an understanding of the source, content and relevance of the information contained within.

I did however set up a more friendly auditing system with my audit.ipynb (iPython Notebook), so one can more easily audit the available tags (448 distinct in austin_texas.osm). Often, more research needs to be performed before one can write code to normalize the tag contents or make decisions about removing or correcting information as mentioned in an example in the first section regarding the “amenity” tag.

5. Referenced Materials

http://docs.mongodb.org/manual/tutorial/getting-started/

http://www.tutorialspoint.com/python/python_reg_expressions.htm

http://okfnlabs.org/blog/2013/10/17/python-guide-for-file-formats.html

http://wiki.openstreetmap.org/wiki/OSM_XML

http://www.austintitle.com/downloads/AustinStreetGuide.pdf

http://discussions.udacity.com/t/project-2-problems-with-trying-to-remove-tiger-gnis-data/13856

http://en.wikipedia.org/wiki/Texas_state_highways

https://www.udacity.com/course/viewer/#!/c-nd002/l-3168208620/m-3189488621

http://overapi.com/python/

https://docs.google.com/document/d/1F0Vs14oNEs2idFJR3C_OPxwS6L0HPliOii-QpbmrMo4/pub

http://en.wikipedia.org/wiki/ZIP_code#ZIP.2B4

http://www.city-data.com/zips/14150.html

https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=nyle%20maxwellcjd%20austin