Cleaning OpenStreetMap Data of Houston

May 30, 2015

1 Introduction

I chose Houston as my studying case, simply because right now I live here. It's the fourth largest city in America and has many immigrants, mostly of Mexican descent. The map data of Houston was obtained from Mapzen, https://s3.amazonaws.com/metro-extracts.mapzen.com/houston_texas.osm.bz2. It's named "houton_texas.osm", with a size of 540M. By sampling with "sampling.py", a sample file, "sample.osm", about 55M, was obtained. However, all operations on the full data take less than 2 minutes and $<$ 200M memory in my laptop, so the sample file was only used for explorations. All discussions below are for "houton_texas.osm". The codes for the sample file are kept but commented out so it's straightforward if you want to switch to the sample file.

2 Cleaning

The cleaning codes are in "cleaning.py". The cleaning was carried out as follows.

2.1 Check Tag Name

First, I counted all tag names and attributes to check for any problematic tag and attribute name, especially for those with few counts. The output was stored in "check_tag_name.md". Everything seems to be right.

2.2 Check v Value

Then I examined attribute v of all tags. After exploration, some attribute k's were chosen and stored in list "tag_names". Their corresponding attribute v values were stored in "v_values_before_cleaning.md". Some problems came out at this step, as discussed below.

2.2.1 Manual cleaning

First of all, there are some typos and misplaced information for sure. Here are three examples.

```
<node ...
    <tag k="addr:street" v="6111 N Ossineke Dr, Spring, TX
        77386" />
    ...
</node>
```

```
<node ...
    <tag k="addr:housenumber" v="1200 East Blvd." />
    ...
</node>
```

```
<node ...
    <tag k="addr:street" v="912" />
    <tag k="addr:housenumber" v="St. Emanuel" />
    ...
</node>
```

In the first one, the full address rather than "N. Ossineke Dr." was put in "addr:street". Similar mistake happened in the second case. "addr:housenumber" was supposed to have value of "1200", without street name. In the third one, the values of "addr:street" and "addr:housenumber" should be interchanged.

These certain mistakes don't have a general pattern. Fortunately, they don't occur too often, so I cleaned them by hand. For the first two types of mistakes, besides correcting the original tags, new tags like "addr:city"and "addr:state" were added when necessary to keep information. All manual cleaning were summarized in "items_manually_cleaned.md". The data after manual cleaning were saved in "houston_texas_manually_cleaned.osm". From now on, all further cleaning will be operated on this file.

2.2.2 addr:city

There are two types of values under tag "addr:city". For instance, "Alvin" and "Alvin, TX", namely, some carry the name of the state. The "TX" or anything similar was removed and only city names were kept.

2.2.3 addr:housenumber

The inconsistency here was that it could be in the form of "111-A", "111 A" or "111 #A". After cleaning, all were changed to the form of "111 #A".

2.2.4 addr:postcode

The problem of postcode was similar to that of "addr:city". Some postcodes were like "TX 77009". "TX" was removed in cleaning. Another problem was that some postcodes had more than five digits. For example, "77005-1890". I didn't change them because extra digits carried information and would be easy to truncate when necessary.

2.2.5 addr:state

"TX", "TX - Texas", "Texas", "Tx" and "tx" all appeared in the data. All were replaced with "TX".

2.2.6 service

"drive-through" was replaced with "drive_through", according to OpenStreetMap's convention.
https://wiki.openstreetmap.org/wiki/Key:drive_through

2.2.7 tiger:county

Three forms of seperator were used under this tag. They were colon: "Austin, TX:Fort Bend, TX", semicolon: "Brazoria, TX;Harris, TX", and semicolon followed by a space: "Fort Bend, TX; Harris, TX". All were changed to colon for consistency.

2.2.8 addr:street

Two main problems were found.

- Incomplete name, for example, "Durham", "Bailey" and "Maroneal". Their full name were found by searching on Google map. After cleaning by function "update_street_name_special", they became "Durham Drive", "Bailey Road" and "Maroneal Street". However, there were still few of them couldn't be located because of the lack of information. I kept them after I had tried my best.

- Over-abbreviated street names, for example, "W Sam Houston Pky N". It was cleaned by function "update_street_name_general" to "West Sam Houston Parkway North". Similar for others.

2.2.9 Recheck v value

After cleaning, the data were written into "cleaned_file.osm" and its v values were rechecked. It was much better.

2.3 Transform to JSON

The elements with name "node" and "way" in cleaned data were transformed to a JSON file, imported into MongoDB. However, a problem came up. The number of nodes plus that of ways didn't equal to the length of the entire file.

After scrutiny, it turned out that in some elements, they got a tag with $k =$ "*type*". For them, during the transformation, key "type" was first assigned to "node" or "way" and later overrode by the child tag with $k =$ "*type*". To solve it, I stored the v value of '$k =$ "*type*" to a new key called "k_type" and redid transformation. The correct JSON data was dumped into "cleaned.json".

3 Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them. The JSON file was mongoimported into db.examples.houston. All queries were saved in "mongodb_query.py" and results in "MongoDB_exploaration.md". They are also shown below for

convenience. Queries use pymongo grammar.

File sizes:

"houton_texas.osm": 541M

"cleaned.json": 607.1M

- Length of the file:

```
> db.houston.count()
2628926
```

- Number of unique users:

```
> len(db.houston.distinct("created.user"))
1053
```

- Number of nodes:

```
> db.houston.find({"type":"node"}).count()
2380486
```

- Number of ways:

```
> db.houston.find({"type":"way"}).count()
248440
```

- Number of cafes:

```
> db.houston.find({"amenity": "cafe"}).count()
72
```

- Number of Starbucks:

```
> db.houston.find({"name": "Starbucks"}).count()
40
```

- Number of Starbucks labeled as cafe:

```
> db.houston.find({"amenity": "cafe", "name": "
    Starbucks"}).count()
40
```

- Top 1 contributing user:

```
> db.houston.aggregate([{'$group':{'_id':'$created.
    user', 'count':{'$sum':1}}}, {'$sort':{'count':
    -1}}, {'$limit': 1}]):
{u'count': 659497, u'_id': u'woodpeck_fixbot'}
```

- Top 10 appearing restaurants:

```
> db.houston.aggregate([{"$match":{"cuisine":{"$exists
    ":1},"amenity":'restaurant'}},{'$group':{'_id':'$
    cuisine', 'count':{'$sum':1}}}, {'$sort':{'count':
    -1}}, {'$limit': 10}]):
{u'count': 47, u'_id': u'mexican'}
{u'count': 28, u'_id': u'american'}
{u'count': 19, u'_id': u'italian'}
{u'count': 15, u'_id': u'pizza'}
{u'count': 15, u'_id': u'burger'}
{u'count': 13, u'_id': u'chinese'}
{u'count': 9, u'_id': u'seafood'}
{u'count': 8, u'_id': u'steak_house'}
{u'count': 7, u'_id': u'barbecue'}
{u'count': 5, u'_id': u'vietnamese'}
```

- Number of ATMs:

```
> db.houston.find({"amenity": "atm"}).count()
12
```

- Number of banks:

```
> db.houston.find({"amenity": "bank"}).count()
188
```

- Top 10 appearing amenities:

```
> db.houston.aggregate([{"$match":{"amenity":{"$exists
    ":1}}},{'$group':{'_id':'$amenity', 'count':{'$sum'
    :1}}}, {'$sort':{'count':-1}}, {'$limit': 10}]):
{u'count': 2485, u'_id': u'parking'}
```

```
{u'count': 2435, u'_id': u'place_of_worship'}
{u'count': 1659, u'_id': u'school'}
{u'count': 669, u'_id': u'fast_food'}
{u'count': 466, u'_id': u'restaurant'}
{u'count': 429, u'_id': u'fountain'}
{u'count': 402, u'_id': u'fire_station'}
{u'count': 327, u'_id': u'fuel'}
{u'count': 209, u'_id': u'grave_yard'}
{u'count': 188, u'_id': u'bank'}
```

4 Discussions

- More than half of cafes reported are Starbucks. They're just so popular.
- The number of ATMs is much less than that of banks, but from daily life experience we know that a bank has at least one ATM, so it can't be true. People are more reluctant to upload information of ATM, probably because they usually don't think of ATMs as being amenities.
- The most popular food is Mexican. Quite fair. We are in Texas!

5 Additional Thoughts

In the data wrangling, the most annoying step was manual cleaning, which was the straightforward solution for those "hard" mistakes, like typos and obviously misplaced information. They could be solved programmatically, like using regex, but the codes would be hardly portable because of the randomness of such mistakes. As the result, I didn't bother to do so. Though such mistakes were only a few but they asked for lots of efforts. It would be a good idea if OpenStreetMap could do some prescreening. For example, they could check for the type of postcode to see if it's a number. A potential problem of prescreening is that there might be some special cases that don't obey the constraints but are still right. As the world is large, it's better to keep some space for special cases. The compromise can be that when an unusual input comes in, a warning would be thrown, but it's not mandatory. It just reminds the user to double-check the input. This simple procedure could significantly reduce unconscious errors, and save data scientist's life.

6 Notes

Some codes are commented out in the final version. To repeat the results above. First you need to extract those zipped files in the directory. Then you can checkout the corresponding commits:
33b38a1310479872bebe116c616a7f39450aa4ca (check tag names)
e6ec0006052a459216499195047fb5cd0286ad44 (check v values)
791cc75287c2ef8645ac32f148d1195de167177b (check v values after cleaning)
72b1b3c88b19a525c88cbf241af25d981822ce4d (write cleaned data to an osm file)
c4dbfe210f2ee42050def47c7382ffcfc9590be7 (write cleaned data to an JSON file).