Project_Submission

August 8, 2017

1 Open StreetMap Case study

1.1 Map Area

Lincoln, Nebraska, United States https://www.openstreetmap.org/relation/169588

1.2 Problems Encountered in the Map

After downloading a openstreet of the lincoln area and narrowing it with SampleOSM.py file, I noticed 3 main problems with the data listed in the following: 1. overabrraviated street name, such as (N Hill Rd) 2. street name end with a comma (6955 Old Cheney Rd,) 3. Data pulled out of GPS 4. Secondary Key is "FIXME" 5. street number starts with a pound sign(Pioneer Woods Drive #110)

I chosed lincoln becasue there is where I spent my 5 years in university, it is the only city in united states that I am familiar with.

1.3 Data cleaning

To solve first 2 problems is not too much a issue by updating varaiable feature and only comma combining with Rd was observed, so I added 'Rd,' into mappling dictionary to substitute the wrong formatted name with the desired one. Following is the code:

```
In [1]: # UPDATE VARIABLEs
        mapping = { "St": "Street",
                    "St.": "Street",
                    "Rd.": "Road",
                    "Ave": "Avenue",
                    "Rd": "Road",
                    "Rd,": "Road"
                    }
```

```
In [2]: # update name method
        def update_name(name, mapping):
            words = name.split()
            for word in words:
                for k in mapping:
                    if word == k:
                        name = re.sub(k, mapping[k], name)
            return name
```

Secondary tag start with "tiger:" confused me at first, before I realized the fact that openstreetmap might be embedded with GPS services. My code identifies anything if that the secondary tag "k" value contains a ":" , the characters before the ":" should be set as the tag type and characters after the ":" should be set as the tag key by using regular expression. In this case, the tag type is "tiger".

```
In [ ]: LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')
        # using regular experssion search tag values containing colon and put them into the co
        elif LOWER_COLON.match(child.attrib['k']):
            tag2['id'] = element.attrib["id"]
            tag2['key'] = child.attrib["k"].split(":", 1)[1]
            tag2['value'] = update_name(child.get('v'), mapping)
            tag2['type'] = child.attrib["k"].split(":", 1)[0]
```

For tags whose secondary Key value is "FIXME", I will just simply ignore them, since they won't contribute anything to the analysis.

A lot of street number starts with a pound during the audit. It is prefered to have number by itself in the street address. In order to do this, using a regex expression is the fastest way to search all the number with a pound sign.

```
In [ ]: find_pound = re.compile(r'\#\d+')
```

Next step is just simply updating our update_name function using regex search and subsitute with the part needed justification.

```
In [ ]: def update_name(name, mapping):
            words = name.split()
            for word in words:
                if find_pound.search(word):
                    name = re.sub(word, word.split('#',1)[1], name)
                for k in mapping:
                    if word == k:
                        name = re.sub(k, mapping[k], name)
            return name
```

1.4 Data Overview

1.4.1 File sizes

lincoln.db 40.1MB
lincoln.osm 69MB
nodes.csv 26.2MB
nodes_tags.csv 1MB
ways.csv 2.2MB
ways_nodes.csv 8.6MB
ways_tags.csv 6.4MB

1.4.2 Number of nodes

```
In [ ]: sqlite> SELECT COUNT(*) FROM nodes;
```

301338

1.4.3 Number of ways

```
In [ ]: sqlite> SELECT COUNT(*) FROM way;
```

35603

1.4.4 Number of unique users

```
In [ ]: sqlite> SELECT COUNT(DISTINCT(combine.uid))
        FROM
        (SELECT uid FROM nodes
         UNION ALL
         SELECT uid FROM way)combine;
```

258

1.4.5 Number of restaurants

```
In [ ]: sqlite> SELECT COUNT(DISTINCT(id))
        FROM node_tags
        WHERE node_tags.value = 'restaurant';
```

109

1.5 Additional Idea

1.5.1 Top 10 contributing users

```
In [ ]: SELECT combine.user, COUNT(*) as num
        FROM
        (SELECT user FROM nodes
         UNION ALL
         SELECT user FROM way)combine
        GROUP BY combine.user
        ORDER BY num DESC
        LIMIT 10;
```

"Your Village Maps",14759
balcoath,78862
woodpeck_fixbot,24287
"James GIS",21858
PHerison,19265
"Stretch Longfellow",7339
KyleD402,5889
bdiscoe,5228
TIGERcnl,2360
JReed,1983

Top user contribution 48.98% Top 2 users contribution 75.15% Most of the data is contributed by "Your Village Maps" and "balcoath".

1.6 Conclusion

In this project, I successfully investigated Openstreetdata of lincoln city by using a iter parser to parse the osm file and convert it into csv file. After importing all the data into data base using sqlite, I find out most of data was contributed by several users and a single GPS provider. To improve a better accuracy of map, more data sources should be introduced into this research such as implementing another GPS provider to provide data nodes. You can map the new data nodes with the old data nodes to validate the data. I also find there is plenty of room to have a more cleaning data after exploring the data from the data base, addresses formatting informally escaped from the cleaning process.

Benifit: 1. improve map awareness 2. trust-worthy data nodes after mapping

Anticipated Problem: 1. more complicated data wrangling process 2. incompliant data at same position