Project-4-Wrangle-OpenStreetMap-Data

Nan Li

Map Area Houston, TX, United States

- OpenStreetMap URL (http://www.openstreetmap.org/search?query=houston#map=11/29.7593/-95.3675)
- MapZen URL (https://mapzen.com/data/metro-extracts/metro/houston_texas/)

Since I'm living and studing in houston right now, I'm more interested to see what's going on here.

1. Sampling Data

Due to the size of the XML file, I sampled the data by using `map_sample.py`. In order to keep more data, set k = 10. And named sample file is `sample_houston.osm`.

2. Data Audit

Unique Tags

Use the `tag_count.py` file to fetch the all tag type with total number of each tag.

- `'member': 1487,`
- `'nd': 309740,`
- `'node': 250482,`
- `'osm': 1,`
- `'relation': 202,`
- `'tag': 180462,`
- `'way': 38132`

Auditing the k Tags

After reading the OpenStreetMap Wiki (https://wiki.openstreetmap.org/wiki/Main_Page) , I learnt that nodes, ways and relations. Because the `"k"` value of each tag contain different patterns, I used

`auditing_k_tag_type.py` including three regular expressions to check for certain patterns in the tags. Four different types of k tags as below.

- `"problemchars" : 0`, for tags with problematic characters, and
- `"lower" : 86870`, for tags that contain only lowercase letters and are valid,
- `"lower_colon" : 89297`, for otherwise valid tags with a colon in their names,
- `"other" : 4295`, for other tags that do not fall into the other three categories.

Auditing the users

Used `auditing_users.py` to audit the users and the number of users. `The number of different users is 1254.`

3. Problems Encountered in the Map

After dealing with the sample file of houston area, I noticed three main problems with the data.

- `Inconsistent street names("Ave", "Frwy")`
- `Inconsistent postal codes("TX 77086", "77024-8022")`
- `Many 'k' tags had only been used once and many similar tags were referenced by different names.`

Inconsistent Street Names

Auditing the street name by using `auditing_street_name.py`, some inconsistent names are showing.

- `'Ave': {'E Parkwood Ave', 'Washington Ave'}`
- `'Blvd': {'John Freeman Blvd', 'Montrose Blvd'}`
- `'Dr': {'Business Center Dr', 'Portway Dr'}`

Updating Street Names

Updating the street name, I used `updating_street_name.py`.

- `('Washington Ave', '=>', 'Washington Avenue')`
- `('Montrose Blvd', '=>', 'Montrose Boulevard')`
- `('Business Center Dr', '=>', 'Business Center Drive')`

Inconsistent Postal Code

Normalized the postal code to be five degits by using `update_zipcode.py`.

- `'TX 77086', '=>' ['77086']`
- `'77077-9998', '=>', '77077'`

4. Data Overview

Preparing Data for SQL

Before using SQL to process data, I convert the OSM to CSV by `xml_to_csv.py`. And then based on the schema from the instruction and `csv_to_db.py`, I create a houston.db file.

File Sizes:

- `houston.osm: 556 MB`
- `sample_houston.osm: 56.4 MB`
- `nodes_csv: 20.2 MB`
- `nodes_tags.csv: 516 KB`
- `ways_csv: 2.20 MB`
- `ways_nodes.csv: 7.31 MB`
- `ways_tags.csv: 5.74 MB`
- `houston.db: 41.5 MB`

Number of documents:

```
sqlite> SELECT COUNT(*) FROM ways_nodes
```

Output:

```
309740
```

Number of nodes:

```
sqlite> SELECT COUNT(*) FROM nodes
```

Output:

```
250482
```

Number of ways:

```
sqlite> SELECT COUNT(*) FROM ways
```

Output:

```
38132
```

Number of unique users:

```
sqlite> SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

Output:

```
1253
```

Number of users contributing only once:

```
sqlite> SELECT COUNT(*)
FROM(SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
HAVING num=1) u;
```

Output:

```
245
```

Top 10 contributors:

```
sqlite> SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num
DESC LIMIT 10;
```

Output:

```
afdreher        46707
woodpeck_fixbot         35168
cammace             19320
scottyc             18677
claysmalley         13835
brianboru         10765
skquinn             8098
RoadGeek_MD99         7674
Memoire             5599
TexasNHD        4673
```

Top 10 zipcode:

```
sqlite> SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
SELECT * FROM ways_tags) tags
WHERE tags.key = 'postcode'
GROUP BY tags.value
ORDER BY count DESC LIMIT 10;
```

Output:

```
77096        47
77449        27
77401        25
77339        22
77494        16
77076        14
77002        11
77586        11
77007        9
77006        8
```

5. Additional Data Exploration

Common ammenities:

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

Output:

```
parking            382
place_of_worship    223
school             148
fast_food          92
restaurant         82
fountain           72
fuel               55
fire_station       30
pharmacy           28
bank               22
```

Popular cuisines:

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC LIMIT    10;
```

Output:

```
mexican            10
burger             5
italian            3
asian              2
pizza              2
american           1
barbeque           1
cajun              1
chinese            1
latin_american         1
```

From this partial result, I can guess there are more mexican who lived in houston since mexico is close to houston.

6. Conclusion

Since the size of the houston.OSM is big, I have played with sample_houston.OSM. There are some inconsistent informations in this data. Doing the normalization is a good way, but it's not the whole idea. If we can standard the input, that's a better way. From the popular cusines, we can know no enough infomation arranged into some tags. Comparing with Google Maps, the information of the dataset were very old.

Additional Suggestion and Ideas

How to control typo errors

- Making some rules to the input data which all the users can follow that. Just like when you order something on Amazon and type your address, it will give you a suggested address.
- Developping some script to clean the data in a certain time.

Benefits

- Avoiding the invalid information on the dataset and improving the fetching result speed and accurancy
- Standarding the input data as Google Maps, it's more convenient to do the cross-referencing.

Anticipated Problems

- Supposed the user still typed the information by unfollowed the suggested format is a big problem.
- For the scrip of cleaning data, it will not only modify the data, but also delete the data sometimes. How should we deal with it?
- Even if everything is setted up exactly, we will still face the wrong information problem. Like someone typed Houston Avenue as Washington Avenue.

References

https://github.com/pratyush19/Udacity-Data-Analyst-Nanodegree/tree/master/P3-OpenStreetMap-Wrangling-with-SQL (https://github.com/pratyush19/Udacity-Data-Analyst-Nanodegree/tree/master/P3-OpenStreetMap-Wrangling-with-SQL)
https://github.com/mablatnik/Wrangle-OpenStreetMap-Data/blob/master/OpenStreetMap.ipynb (https://github.com/mablatnik/Wrangle-OpenStreetMap-Data/blob/master/OpenStreetMap.ipynb)