Open Street Map Case Study

Map Area

Dallas, Texas, United States

- https://www.openstreetmap.org/relation/6571629#map=10/32.8188/-96.7321

Dallas is a well-known city near to my town. So, I think it's my best opportunity develop the map in clear view. But the validation is a long-time process to do so, here we are taking a small area Plano from the city to validate the map. The link of Plano is:

- https://mapzen.com/data/metro-extracts/your-extracts/d49109655fb8

Above mentioned are the links to download the map. In the city Dallas, the Unique Tags are:

1) 'node': 5960064, 2) 'nd': 6720721,3) 'bounds': 1,4) 'member': 31975, 5) 'tag': 3102167,

6) 'relation': 3108, 7) 'way': 619870, 8)'osm': 1

Which are counted using the file 'mapraser.py', explains the 619870 ways of a 1.25GB osm file with 3108 relations for city Dallas.

Different Tags:

1. 'lower': 1085496: these tags contain valid lower case letters in the data.
2. 'lower_colon': 1977083: these contains colon in their names and the letters are in lower case which are valid in the data.
3. 'other': 39581: these not come under those 3 categories.
4. 'problemchars': 7 : the tags contains problematic charecters.

1. Problems Encountered in the Map

DATA AUDIT:

After downloading the sample city data and running the 'p3_projectaudit.py'. I found there are some problems encountered as shown.

- The street names are written in a short-forms (*v="Chase Oaks Blvd"*).
- The city name is written in lower case letter (*v=" dallas"*)
- Incorrect phone numbers (some the phone numbers not clearly written "*v=" v="+1 972.234.2390"/>""*.(dots should be taken off)
- Some data is dragged form the Tiger GPS and most of their street names are not mentioned, shown in the following format:
  <tag k="tiger:cfcc" v="A41"/>
  <tag k="tiger:county" v="Dallas, TX"/>
  <tag k="tiger:reviewed" v="no"/>

And in these project, we are cleaning and updating two major problems encountered in the map they as follow:

Over abbreviating the street names:

In p3project_audit.py the street names are cleaned and over abbreviated in to original names and they are uploaded in a csv files by calling the function *'update_names(names)'* in the p3_projectdatacovert.py file, the function used to clean the names is:

def update_name(name):

```
update = []
   for splitting  in name.split(' '):
      if splitting in mapping.keys():
         splitting = mapping[splitting]
      update.append(splitting)
   return " ".join(update)
```

Here name is variable taken from the sample osm file where there is a street name and it splits the name verifies matching word from the mapping in audit file, if there is any it updates the name by choosing the perfect one which matches to the word. Some of them as shown.

```
(u'West Campbell Road', 23), #here rd => Road, Dr => Drive, Dr. =>Drive are updated.
 (u'Rockingham Way', 18),
 (u'Courtside Lane', 17),
 (u'Nightfall Drive', 17),
 (u'Donnington Drive', 16),
 (u'Lantern Light Drive', 16)
```

Phone Numbers: -

In the raw data, the phone numbers and their area codes mentioned in uneven manner. So, they are cleaned by separating the area codes of the number and updated to the csv files by calling the function from p3project_audit.py file.

```
Sqlite>SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM new_nodes_tagss UNION ALL
 SELECT * FROM ways_tags) tags
 WHERE tags.key='phone' GROUP BY tags.value  ORDER BY count DESC LIMIT 10
```

Here are the 10 results of the phone numbers:

```
[(u'(214) 484-5211', 1),
 (u'(214) 620-0630', 1),
 (u'(214) 888-7771', 1),
 (u'(469) 200-0972', 1),
 (u'(903) 413-9405', 1),
 (u'(972) 231-2695', 1),
 (u'(972) 231-6344', 1),
 (u'(972) 234-4391', 1),
 (u'(972) 234-4697', 1),
 (u'(972) 234-5300', 1)]
````

The three numbers in the brackets are the area codes of the number. Which are kept in a sequence order now by removing all dirty data like dots. And the validation is a time taking process so, the validation is done only for the sample osm file i.e, Plano.

Cities: -

There are some other cities and there are down loaded with the dirty data they are:

```
Sqlite>SELECT tags.value, COUNT(*) as count FROM
(SELECT * FROM new_nodes_tagss UNION ALL
SELECT * FROM ways_tags) tags
```

```
WHERE tags.key LIKE '%city'
 GROUP BY tags.value ORER BY count DESC;
```

The results are as shown:

```
[(u'Plano', 235),
 (u'Richardson', 68),
 (u'Garland', 8),
 (u'100', 1),
 (u'5', 1),
 (u'Allen', 1),
 (u'Allen TX', 1),
 (u'Plano, TX', 1),
 (u'richardson', 1)]
```

So, we can say that the phone numbers and the postal codes are sown in the data not only belongs to the ci ty Plano there are other cities as shown and the luckily, we can say that the phone numbers are given with their area codes and each phone number is used only on time.

2. Data Overview

Data Overview and Additional Ideas:

Files and their sizes:

`p3plano_dallas_.osm`: 23.4 MB, `p3plano_dallas_.db` : 16.8 MB, `p3_projectaudit.py` 4.0 KB,

`p3_databaseconvert.py`: 8.0 KB, `p3_project_databasetables.py`: 9.0 KB, `README.md`: 8.0 KB

`p3project.pdf`:344 KB, `schema.py` : 3KB, `nodes.csv`: 8.84 MB, `nodes_tags.csv`: 174 KB

`ways.csv`: 793 KB, `ways_nodes.csv`: 2.93 MB, `ways_tags.csv`: 2.54 MB

Number of Nodes:

```Sqlite>SELECT COUNT(*) FROM new_node_table```

```
no.of nodes: 105182
```

Number of ways:

``` Sqlite>SELECT COUNT(*) FROM new_wayss```

```
no.of ways: 12911
```

Number of unique users:

```
Sqlite>SELECT COUNT(DISTINCT(i.uid))
FROM (SELECT uid FROM new_node_table
UNION ALL SELECT uid FROM new_wayss) i;
```

```
[(299,)]
```

Top 5 Contributing users:

```
Sqlite>SELECT i.user, COUNT(*) as num
FROM (SELECT user FROM new_node_table
UNION ALL SELECT user FROM new_wayss) i
 GROUP BY i.user ORDER BY num DESC
LIMIT 5
```

```
[(u'woodpeck_fixbot', 31364),
```

```
 (u'Andrew Matheny_import', 24513),
 (u'Stephen214', 10191),
 (u'Eagle1295', 3836),
 (u'stangoodman', 3462)]
```

Users having only one post:

```
Sqlite>SELECT COUNT(*) FROM (SELECT i.user, COUNT(*) as num
FROM (SELECT user FROM new_node_table UNION ALL SELECT user FROM new_wayss) i GROUP BY
i.user HAVING num=1)  u;
```

```
[(51,)]
```

Postal Codes:

```
Sqlite>SELECT tags.value, COUNT(*) as count FROM
(SELECT * FROM new_nodes_tagss
 UNION ALL  SELECT * FROM ways_tags) tags
 WHERE tags.key LIKE '%postcode'
   GROUP BY tags.value
  ORDER BY count DESC;
```

```
[(u'75080', 53),
 (u'75075', 22),
 (u'75074', 19),
 (u'75023', 15),
 (u'75081', 7),
 (u'75044', 6),
 (u'75082', 6),
 (u'75025', 4),
 (u'75040', 4),
 (u'75024', 3),
 (u'75041', 2),
 (u'75080-3338', 2),
 (u'75002', 1),
 (u'75013', 1),
 (u'75054', 1),
 (u'75093', 1),
 (u'75248', 1),
 (u'75252', 1),
 (u'TX 75074', 1)]
```

Additional Ideas and gamification Suggestions: -

Due to the manual editing and automated changes in the map there is a drastic change in the contribution of users as shown above, Some statistics about the users.

- The top user is `'woodpeck_fixbot'` having a more than half contribution.
- The top two users are contribution is near by 80%.
- And 51 users are having only one post.

1. By these usages of users concludes to recommendation of **'gamification'** as a motivation force for user contribution such as rewards, badges.
2. Some restrictions should be made when entering their data like using special characters.
3. The data should be cleaned periodically by taking their changes there addresses and phone numbers.

4. And when we are logging in to site it is taking the current location details and auto filled but coming to the postal code it is showing the present location and all other city codes which are not near me too and some of them are with 3 or 6 digits which are wrongly shown. So, it should provide an online survey like google and Wikipedia which is not providing now to select the postal code.

More Information(benefits): -

1. The most users should be encouraged to enter the motivated data more information which improves the views like as we see in the Google maps, Go Pokeman. So, that it will be user friendly.
2. All these ideas make the people to survey easily and make the decisions for their purposes like Business men search for an area where the competitors are less for him.
3. The views are increased by seeing the information about popular places and they can use phone numbers to make sure.

3. Additional Data Exploration

Top 20 amenities in the data:

```
Sqlite>SELECT value, COUNT(*) as num FROM new_nodes_tagss WHERE key='amenity' GROUP BY value
ORDER BY num DESC LIMIT 20;
```

```
[(u'restaurant', 83),
 (u'fast_food', 41),
 (u'place_of_worship', 25),
 (u'fuel', 23),
 (u'cafe', 16),
 (u'post_box', 14),
 (u'bank', 12),
 (u'bicycle_repair_station', 8),
 (u'fountain', 8),
 (u'library', 6),
 (u'fire_station', 5),
 (u'pharmacy', 5),
 (u'hospital', 4),
 (u'parking', 4),
 (u'school', 4),
 (u'doctors', 3),
 (u'bench', 2),
 (u'college', 2),
 (u'toilets', 2),
 (u'bar', 1)]
```

Top two religions:

```
Sqlite>SELECT new_nodes_tagss.value, COUNT(*) as num FROM new_nodes_tagss JOIN
                (SELECT DISTINCT(id) FROM new_nodes_tagss
                  WHERE value='place_of_worship') i
                 ON new_nodes_tagss.id=i.id WHERE new_nodes_tagss.key='religion'
                 GROUP BY new_nodes_tagss.value
                ORDER BY num DESC LIMIT 2;
```

```
[(u'christian', 23), (u'unitarian_universalist', 1)]
```

Popular Cuisines: -

```
```

```
Sqlite>SELECT new_nodes_tagss.value, COUNT(*) as num FROM new_nodes_tagss
            JOIN (SELECT DISTINCT(id) FROM new_nodes_tagss WHERE value='restaurant') i
            ON new_nodes_tagss.id=i.id WHERE new_nodes_tagss.key='cuisine'
            GROUP BY new_nodes_tagss.value ORDER BY num DESC;
```
```
[(u'mexican', 8),
 (u'chinese', 5),
 (u'pizza', 5),
 (u'italian', 4),
 (u'american', 3),
 (u'burger', 3),
 (u'ice_cream', 3),
 (u'regional', 3),
 (u'sushi', 3),
 (u'chicken', 2),
 (u'sandwich', 2),
 (u'Vietnamese', 1),
 (u'asian', 1),
 (u'breakfast', 1),
 (u'french', 1),
 (u'indian', 1),
 (u'international', 1),
 (u'japanese', 1),
 (u'mexican;italian;casual dining', 1),
 (u'salads', 1),
 (u'seafood', 1),
 (u'taco_restaurant', 1),
 (u'thai', 1),
 (u'wok', 1)]
```

4. Conclusion

The map is known is clear to understand here in the projects in the first the raw data is taken in a .osm file and it cleaned and audited the names to get better view in the map. The validation is time taking process to convert the data in to csv files and in the osm file the data taken from the tiger gps which improperly written data which is cleaned now and I think the data can be updated to Openstreetmap.org now for better view for the future users and to make it more popular.

Files:

`p3 plano_dallas_.osm` : It is a sample raw data downloaded from the link for the wrangling and validation purpose.

`p3 projectaudit .py`: Used to audit, clean and updating files with separate sections(cells) in single file.

`P3 databaseconvert.py`: Used to make the csv files.

`p3 project databasetables.py`: used to queries using the database sqlite and additional data explosion

`p3mappraser.py`: It find no.of unique tags of the data.

`p3tags.py`: different type of tags and there counts.

`p3project.pdf`: This file.

`READM.md`: Documentation for the code.