Project 1: Wrangle OpenStreetMap Data

Name: Wandi Cui
Map Area: I chose Los Angeles because I am a student at UCLA now and I want to know more about the place where I study. However, the whole data of Los Angeles is around 8G, so I only extracted an area with the minimum latitude 34.0412806, minimum longitude -118.2485964, maximum latitude 34.1186262 and maximum longitude -118.4600828 respectively.

- Location: Los Angeles, CA, USA
- MapZen URL: https://mapzen.com/data/metro-extracts/your-extracts/0b2605ecf6fa

1. Problems Encountered in the Map

1.1 Data Statistics

Observing the XML file, I found that there are several different kinds of tags, so I parse the dataset with cElementTree module of Python and count the numbers of unique tags.

statistic.py is used to do some simple statistic calculation including counting the numbers of unique tags, unique users and tags with different patterns (show more details later).

- 'osm': 1
- 'bounds': 1
- 'node': 2243524
- 'tag': 1348452
- 'way': 203368
- 'nd': 2474587
- 'relation': 4120
- 'member': 14616

I found that the value “k” in each tag have different patterns, in order to extract information correctly and reasonably in the phase of transferring XML file to CSV file, I separated it into 4 categories and designed 3 regular expressions to match the pattern. The 4 tag categories and its number of occurrences are as follows:

- “lower”: 785488, representing valid tags that only contain lowercase letters.
- 'lower_colon': 558847, representing valid tags with a colon in “k” value.
- 'problemchars': 1, representing tags with unexpected characters like “=, +, &, <, > ', “, ?, %, #, $, @” and so on.
- 'other': 4116, representing tags that belong to no categories mentioned above.

Unique number of user: 763.

1.2 Data Audit

Street Address Inconsistencies

For street map dataset, the main problem I found is the street name inconsistencies. Some users may use the full name of streets while others may use abbreviations. Some street names are recorded in lowercase letter while others may not.

To find different street types in dataset, I created a function **audit_street_type()** in **audit.py**, checking elements “node” and “way” iteratively. If their element “tag” contains attribute k with the value of “addr:street”, returns the last word of the “value”. In this way, I got 53 different results which includes some other strings that cannot be a street type. **The result is:** {'#2004', 'Plaza', 'Mansfield', '246-0756', '308', 'Stars', 'West', '#1314', 'North', 'Road', 'Penthouse', 'Avenue', 'Ave', 'Dr', 'Broadway', 'Way', '777-5877', 'Str', '#940', 'Lane', 'Figueroa', 'floor', 'Blvd.', '#1801', 'Thibault', 'Boulevard', 'blvd', 'Rd', 'USA', 'Drive', '#A', 'St', 'CA', 'Circle', '4045', 'Sepulveda', 'Fl', 'East', 'South', 'Place', '#17', 'St.', 'Terrace', '110402', '858-1383', 'Trail', '276-1562', '#C', 'Blvd', '200', 'Street', 'E', 'avenue'}.

Some of the results containing numbers which must be associated with postcode. I ignored those scenarios but found out real street type and its different expressions. Then I created a dictionary to save the mapping relationship of different names for a same street type, which was applied to standardize the tags in dataset **The dictionary is :** street_type_abbr_map = {"Dr": "Drive", "Ave": "Avenue", "avenue": "Avenue", "St": "Street", "Str": "Street", "St.": "Street", "Blvd": "Boulevard", "blvd": "Boulevard", "Rd": "Road", "Fl": "Floor", "floor": "Floor", "Ct": "Court", "Hwy": "Highway", "Pkwy": "Parkway", "Pky": "Parkway", "S": "South", "W": "West", "N": "North", "E": "East"}

Postcode Inconsistencies

Another problem I found is the postcode inconsistencies. I found three formats of postcode in “addr:postcode” area:

- 9****
- CA 9****, such as CA 90036

- 9****-****, such as 90026-3148

There are 6 records of postcode with the second format in the dataset. In addition, there are 33 and 25 records in “nodes” and “ways” element respectively using the last format. Fortunately, I did not find any invalid postcode which does not belong to Los Angeles; so only unifying postcodes into the first simple format is needed.

1.3 Problem Solving

To solve the problems of the dataset mentioned above, I used pandas package in Python. File **update_csv.py** was created to finish this task. Before this, I created file **xml_to_csv.py** to transfer the xml file into five csv files: nodes.csv, ways.csv, nodes_tags.csv, ways_tags.csv, ways_nodes.csv. In this way, all information was extracted and represented structurally and clearly in csv files.

In **update_csv.py**, file nodes_tags.csv and ways_tags.csv were read into two different dataframes. Function **street_update()** updated the street names into standardized pattern, while function **postcode_update()** unified postcode into the simplest format “5 digit” format.

2. Data Overview

File sizes:

- MyLA3.osm: 507.9 MB
- nodes.csv: 208.6 MB
- nodes_tags_update.csv: 1.3 MB
- ways.csv: 13.8 MB
- ways_nodes.csv: 60.2 MB
- ways_tags_update.csv: 50.7 MB
- udacity_project1_final.db: 369.2 MB

Number of nodes:

```
sqlite> SELECT COUNT(*) FROM nodes
   ...> go
2243524
```

Number of ways:

```
sqlite> SELECT COUNT(*) FROM ways
   ...> go
203368
```

Number of unique users:

```
sqlite> SELECT COUNT(DISTINCT(all_user.uid))
   ...> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) all_user
   ...> go
 861
```

Top10 contributing users:

```
sqlite> SELECT all_user.user, COUNT(*) as num
   ...> FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) all_user
   ...> GROUP BY all_user.user
   ...> ORDER BY num DESC
   ...> LIMIT 10
   ...> go
schleuss_imports|557240
manings_labuildings|234168
joemfox_imports|226278
ridixcr_import|209909
karitotp_labuildings|156284
calfarome_labuilding|145441
Luis36995_labuildings|131529
dannykath_labuildings|125774
emamd_imports|70546
schleuss|68400
```

Number of users contributing only once:

```
sqlite> SELECT COUNT(*) FROM
   ...> (SELECT all_user.user, COUNT(*) as num
   ...> FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) all_user
   ...> GROUP BY all_user.user
   ...> HAVING num = 1)
   ...> go
 236
```

Popular Cuisines:

```
SELECT nodes_tags.value, COUNT (*) as num
FROM nodes_tags
        JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = "restaurant") re
        ON nodes_tags.id = re.id
WHERE nodes_tags.key = "cuisine"
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 5
```

OUTPUT: [('mexican', 24), ('pizza', 22), ('american', 20), ('italian', 20), ('thai', 16)]

Common Amenities:

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

OUTPUT: [('restaurant',494), ('place_of_worship', 178), ('cafe', 168), ('fast_food', 148), ('school', 119), ('parking_entrance', 81), ('bicycle_parking', 66), ('waste_basket', 64), ('bank', 61), ('bar', 54)]

3. Additional Data Exploration

Number of One-way Street:

SELECT COUNT(DISTINCT(id)) FROM ways_tags WHERE key = "oneway" and value = "yes"
OUTPUT: 2215

Number of Building or Street that Started to be Built in Different Century:

SELECT COUNT(DISTINCT(id)) FROM ways_tags
WHERE key = "start_date" and (value > "1800" and value <= "2017")
OUTPUT: 162877
SELECT COUNT(DISTINCT(id)) FROM ways_tags
WHERE key = "start_date" and (value >= "1900" and value < "2000")
OUTPUT: 158608
Similarly, 1122 started in 19th century and 3147 started in 21st century. So, most of buildings and streets were built in 20th century with the 97 percent of the total number.

Tourism Categories:

SELECT DISTINCT (value) as tour_type, COUNT (*) as num
FROM (SELECT id, value FROM nodes_tags
WHERE key = "tourism")
GROUP BY tour_type
ORDER BY num DESC
OUTPUT: [('attraction', 90), ('hotel', 48), ('artwork', 34), ('museum', 19), ('motel', 13), ('viewpoint', 10), ('guest_house', 7), ('gallery', 5), ('hostel', 5), ('apartment', 2), ('information', 2), ('picnic_site', 2), ('yes', 2)]

Percentage of wheelchair accessible nodes:

```
sqlite> SELECT COUNT(*) FROM nodes_tags
   ...> WHERE key = "wheelchair" and value = "yes"
   ...> go
 68
sqlite> SELECT COUNT(*) FROM nodes_tags
   ...> WHERE key = "wheelchair"
   ...> go
 81
```

68 / 81 = 84.0%

4. Other Ideas About the Datasets

The OpenStreetMap data of Los Angeles is of reasonable quality, but there are still a variety of improvements needed in the dataset.

- The pattern of nodes with similar category of information is different, making it hard to discover valid information with simple query and statistic methods.
  # Solution: First, a good way to unify the pattern is to offer contributed users a specific format to follow. However, this could only make future data organized, we still have to deal with current data. A way to do so is to transfer original osm file into certain xml format that we expected.
  # Benefits: After this improvement, we can easily process the data with similar category of information without any unexpected results. There will be no missing results caused by different expression of tags and values.
  # Anticipated Issues: Setting fixed pattern of each different node can be complicated since situation is different for each spot in map. We need to observe original dataset to decide which is necessary information and what attributes may need for each category of node (e.g. building or way).
- More additional information for buildings and ways in campus is needed. Observing the dataset, I found that some buildings and ways in UCLA campus; however, they only contain information of themselves without connected with information of the campus.
  # Solution: Most of school provides its campus map on official website, such as: UCLA campus map: http://maps.ucla.edu/downloads/, Some school district information: https://schooldirectory.lausd.net/schooldirectory/.
  With the help of the websites mentioned above and other map website like Google Map, we can get comprehensive campus map information. Then, create tags showing school information (e.g. school name) for those buildings and streets inside campus.
  # Benefits: After this improvement, we can analyze data within each campus and easily recognize buildings and ways in a campus.
  # Anticipated Issues: This task is time-consuming because we need to search the whole dataset to find nodes of buildings and streets in campus and add information for them.