OpenStreetMap Data Wrangling with SQL

Lee Anna

Map Area: Las Vegas, Nevada, USA:

https://www.openstreetmap.org/relation/170117

https://mapzen.com/data/metro-extracts/metro/las-vegas_nevada/

Project Overview

Using data munging techniques to clean the OpenStreetMap data for a part of the world in **https://www.openstreetmap.org**.

Objectives:

- Assess the quality of the data for validity, accuracy, completeness, consistency and uniformity.
- Parse and gather data from popular file formats such as .csv, and .xml
- Process data from multiple files or very large files that can be cleaned programmatically.
- Learn how to store, query, and aggregate data using SQL.

Reason for selection:

- Las Vegas, Nevada: I intended to choose where I have been to and this area is 222MB in size which was the smallest among many options. I would like to analyze my hometown or other metropolitan area I have visited before but I believe this would be a great start.

Auditing data

Problems:

There were several issues with the original dataset as highlighted below.

- Over-abbreviated street type e.g. 'Corporate Park Dr' for 'Corporate Park Drive'
- Inconsistent postal codes e.g. NV 89052, 89108-7049, Nevada 89113
- Incorrect postal codes e.g. 6451112, 8929
- Inconsistent city names e.g. Las Vegas, LAS VEGAS, Las Vegas NV, Las Vagas

Street types:

In **audit_street_name.py**, I print out uncleaned street names by using "pprint" module so that I can refer to the list to fix the mapping variables. E.g. mapping = {"Blvd": "Boulevard", "Cir": "Circle"}.

```
def update_name(name, mapping, street_type_re):
    m = street_type_re.search(name)
    if m:
        st_type = m.group()
        if st_type in mapping:
            name = re.sub(street_type_re, mapping[st_type], name)
    return name
```

After changing the variables, the update_name function corrects the problematic street names to their respective mappings when it runs and the part of outcome is as follows.

```
oval drive => oval Drive
Paradise Rd => Paradise Road
W Warm Springs Rd => W Warm Springs Road
Losee Rd => Losee Road
El Camino Rd => El Camino Road
W Craig Rd => W Craig Road
S Fort Apache Rd => S Fort Apache Road
S Pecos Rd => S Pecos Road
Hillpointe Rd => Hillpointe Road
```

Postal codes:

In **audit_postal_code.py**, I use "regular expression" to get 5 valid numbers in each postal code to make them consistent.

```
zip_type_re = re.compile(r'\d{5}-??')
def update_zip(zip):
    m = zip_type_re.search(zip)
    if m:
        return m.group()
    else:
        return "unknown"
```

And the part of the result is shown below.

```
89014 => 89014
8929 => unknown
89156 => 89156
89131 => 89131
89178 => 89178
89179 => 89179
6451112 => 64511
89134 => 89134
89130 => 89130
NV 89124 => 89124
89135 => 89135
```

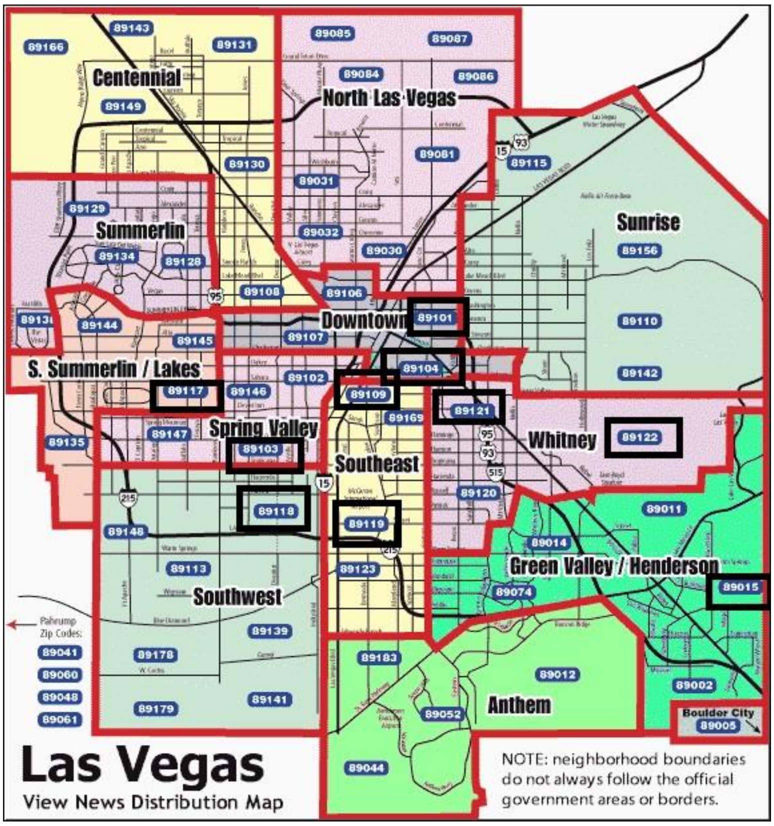
After standardizing inconsistent postcodes, the top 10 count results became more reliable.

```
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;
```

Descending numerical order by counts

```
value|count
89109|104
89101|74
89122|60
89119|56
89117|47
89015|45
89104|34
89121|34
89103|33
89118|31
```

Las Vegas Zip Codes



Reference: city-data.com/forum/las-vegas/

The outcome has revealed that a large part of the data was collected from the middle of Las Vegas. Henderson is the second largest city in Nevada and appeared in the top 10.

City names:

To support the above analysis, I sorted cities by count.

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key LIKE '%city'
GROUP BY tags.value
ORDER BY count DESC
Limit 10;
```

```
Las Vegas|522
Henderson|111
North Las Vegas|29
Boulder City|13
Paradise|13
12|7
15|5
8|5
Las Vegas, NV|5
Overton|5
```

Even though city names are not very consistent, “Las Vegas” is written in an overwhelming number of tags and “Henderson” has the second largest counts.

Data Analysis

<u>Data Overview:</u>

Below is a basic statistical overview of the dataset.

- File sizes

```
las-vegas_nevada.osm ......... 222 MB
las-vegas_nevada.db .......... 124 MB
nodes.csv ............. 87 MB
nodes_tags.csv ........ 2 MB
ways.csv .............. 7 MB
ways_tags.csv ......... 15 MB
ways_nodes.cv ......... 31 MB
```

- Key types in tags (**tags.py**)

```
{'lower': 318077, 'lower_colon': 165530, 'other': 7312, 'problemchars': 0}
```

- Number of nodes

```sql
SELECT COUNT(*) FROM nodes;
```

1052739

- Number of ways

```sql
SELECT COUNT(*) FROM ways;
```

112602

- Number of unique users

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

1067

- Top 10 contributing users

```sql
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;
```

```
alimamo|251482
tomthepom|121167
woodpeck_fixbot|70788
alecdhuse|66523
abellao|55651
gMitchellD|44620
robgeb|40928
nmixter|40029
TheDutchMan13|39078
Tom_Holland|32927
```

- Number of users appearing only one (having 1 post)

```
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;
```

261

Additional Data Exploration:

- Top 10 amenitites: **Restaurant**

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

```
restaurant|477
place_of_worship|294
fuel|282
fast_food|278
fountain|266
school|206
shelter|122
toilets|86
bar|80
cafe|79
```

- Biggest religion: **Christian**

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
   JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='place_of_worship') i
   ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;
```

```
christian|274
```

- Most popular cuisines: **Mexican**

```
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;
```

```
mexican|41
pizza|31
american|20
italian|18
steak_house|16
burger|15
```

```
chinese|13
asian|9
japanese|9
buffet|7
```

Conclusion

- Final thought: Even though many inconsistency and inaccuracy was seen in the dataset from the auditing process, there were not much problematic characters or language problems, which saved the researcher a lot of work.
- Limitations: As long as OpenStreetMap allows volunteers to edit objects such as restaurants and schools, human input errors are inevitable. Furthermore, manual data cleaning process was harder for ambiguity, which was the big problem especially for city names in this report e.g. city name: 12.
- Suggestions: To bring more volunteers and encourage them to input clean data, there are several approaches I would recommend
    1. **Data auditing tools**: Those tools would be a great help to increase the level of data quality. They should carry out consistency, completeness, format, spelling and grammar check when users type in the map contents.
    2. Action: Validation check methods need following actions.

        ① **Advisory action**: It should indicate participants that there is a validation issue with their input and show where the problem is detected.

        ② **Verification action**: It parses every data and restricts invalid input such as foreign language or special characters. It can show recommended options to users when they are entering data e.g. frequently written word.
    3. **Gamification**: It is to make the website look more interesting and fun. Entertaining activities are involved. The detailed examples are ranking system, community context, evaluation and reward systems. They can motivate and challenge high performers to add more new data and correct the existing information. Visualizing their contribution is essential in this case.
    4. **Behavioral approaches**: OpenStreetMap should be well-organized to increase the chance to see the map for volunteers. There should be no barriers to their contents input. It can also send them notifications of where should be updated or if it has found a street sign on the area they seem interested.
- Benefits: Suggestions mentioned above can lead to number of users increase and higher satisfaction among people who got rewarded. Furthermore, the higher data quality OpenStreetMap achieves, the more likely it will appeal to potential investors.
- Anticipated issues: First, it would cost much to implement gamification. Second is that keen competition may cause an accuracy problem because a few contributors might input fake data. In this case, cross referencing validation will be needed. Third, volunteer retention will be tough if validation check is too strict. Lastly, if the company focuses too much on gamification, the purpose of the project will become less important and faded. Volunteers will forget what they were willing to do.

References

http://www.city-data.com/forum/las-vegas/487322-neighborhoods-overview-map-zip-codes.html

https://www.openstreetmap.org/relation/170117

https://mapzen.com/data/metro-extracts/metro/las-vegas_nevada/

https://en.wikipedia.org/wiki/OpenStreetMap