OpenStreetMap Data Case Study - Las Vegas

Map Area

Las Vegas, NV, United States

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

This map is one of my favorite city in the US. The restaurants, the casinos and the hotels always attract me. I would like to explore more though this Open Street Map project.

Problems Encountered in the Map

The original data set is 192MB and I am luck enough to have a fast enough computer that is able to process large amount data. So I am able to go through the **entire** data to find any potential problems that I may encounter in this project.

- Inconsistent street name format, *(St. vs Street, Ave vs Avenue, and Blvd. vs Boulevard etc)*
- Inconsistent postal codes *("89144" vs "89144-XXXX", "NV 89144")*
- None ASCII code attributes or certain characters that shouldn't exist in certain names.

Inconsistent street name format

To make sure the Suffix of street names follow the same format and convention, I created a mapping as following. This ensured that the same street always have the same name. This is useful when we explore the popularity of a certain street using SQL queries, *count()* function can provide the correct answer.

```
ROAD_NAME_MAPPING = {"St": "Street",
            "St.": "Street",
            "Ave": "Avenue",
            "Ave.": "Avenue",
            "ave": "Avenue",
            "AVE": "Avenue",
            "Rd.": "Road",
            "Rd": "Road",
            "Blvd.":"Boulevard",
            "Blvd":"Boulevard",
            "blvd":"Boulevard",
            "blvd.": "Boulevard",
            "N":"North",
            "S":"South",
            "S.": "South",
            "Dr":"Drive",
            "Dr.":"Drive",
            "Pkwy":"Parkway",
            "apache":"Apache",
            "Ln.":"Lane"
                     }
```

The following converting function is performed whenever a street name is encountered.

```
    if tag_temp_key == 'street':
        m = street_type_re.search(tag_temp_value)
        if m:
            street_type = m.group()
            if street_type in ROAD_NAME_MAPPING:
                tag_temp_value = tag_temp_value.replace(street_type,
ROAD_NAME_MAPPING[street_type])
```

This updated all substrings in problematic address strings, such that:
“S Las Vegas Blvd”
becomes:
“South Las Vegas Boulevard”

Postal Codes

There are three types of postal code formats in the OSM file.
1. 5 digits numbers: 89144
2. State + 5 digits: NV 89144
3. 5 digits + 4 digits: 89144-XXXX
Type 2 and 3 are converted to type 1 to ensure data consistency.

```
    elif tag_temp_key == 'postcode':
        if tag_temp_value.split(' ')[0] != tag_temp_value:
            tag_temp_value = tag_temp_value.split(' ')[1]
        tag_temp_value = tag_temp_value.split('-')[0]

        if len(tag_temp_value) != 5 or tag_temp_value.isdigit() == False:
            return -1
    return tag_temp_value
```

Problematic Characters

Some non-ASCII chars have been found in the OSM file, such as those from Chinese or Japanese. Assuming the main target readers of this project use English as the only official language, have these chars may impact the readability. Other the other hand, some special chars shouldn't be used in certain names, for example, ?, #, @ should not appear in an address. Therefore attributes with these non-ASCII chars or *problematic chars* are considered as **invalid**. The following code is used to removed records containing these chars.

```
def is_ascii(s):
    return all(ord(c) < 128 for c in s)
```

```
PROBLEMCHARS = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

    if problem_chars.search(child.attrib['v']) != None\
        or is_ascii(child.attrib['v']) != True:
        return {}
```

Data Overview

This section contains basic statistics about the dataset, the SQL queries used to gather them.

File sizes

```
las-vegas_nevada.osm ......... 192 MB
OSM.sqlite .......... 102 MB
nodes.csv ............. 71 MB
nodes_tags.csv ........ 1.6 MB
ways.csv .............. 5.5 MB
ways_tags.csv ......... 12.1 MB
ways_nodes.cv ......... 24.9 MB
```

Number of nodes

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

858919

Number of ways

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

95925

Number of unique users

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

874

Top 5 contributing users

```
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;
```

```
alimamo                 253788
woodpeck_fixbot         74923
alecdhuse               66723
abellao                 56627
gMitchellD              47208
```

Number of users appearing only once (having 1 post)

```
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;
```

189

Insight

This section contains more insight into the dataset, the SQL queries used to gather them.

Top 5 appearing amenities

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 5;
```

```
place_of_worship        314
restaurant              284
fountain                266
fast_food               256
school                  241
```

Top 3 religions

```
sqlite> 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 3;
```

```
christian                294
bahai                    3
jewish                   3
```

Being the largest religion in the US, Christian is top-ranked in Vegas as well. Second and third are Bahai and Jewish, which are much smaller religious group compared to Christian.

10 Most 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;
```

```
mexican                  26
pizza                    24
american                 13
burger                   10
chinese                  9
italian                  8
japanese                 8
thai                     7
asian                    4
sushi                    3
```

The popularity of Mexican food indicates a relative large percentage of Hispanic people in the city. Data from 2010 census indicates 31.5% of Hispanic among the population. Asian food(Chinese, Japanese, Asian, Thai and Sushi) also takes 5 slots in the top 10 rank. However, only 6.1% is shown in 2010's censorship. This is possible because there are good amount of Asian tourist, but not residences.

Data Source

Conclusion and Limitation

This project firstly cleaned up the data by filtering out invalid entries and converting data into consistent format. The data is then written into separated CSV files with more organized format. A SQL databased is use to store the data. Queries are made to provide more detailed information and insight.

Here are some limitation of this data analysis project:

- Data accuracy and completeness is not examined in this project since verifying the data required comparison against real world data.
- The boundary of the selected data is a rectangle which is not usually how a city is defined.
- GPS data is not processed in this project due to the time limitation

Improving the Data Quality of OSM

- Design a more robust front end webpage to screen out invalid data, such as post-code in wrong format, non-ASCII chars, and inconsistence street format etc.
    - Potential Issue: Design a good front end webpage could be labor intensive, which requires large amount of coding. As a project mainly driven by a community of mappers, this could be difficult to implement.
- Encourage user to input more entries by providing virtual rewards, such as medals, badges or higher levels.
    - Potential Issue: The rewards have to be encoring enough to be effective. On the other hand, there is a risk that users may forge fake entries to win rewards which will affect the data quality.
- Cross validating data within the database to ensure data consistency
    - Potential Issue: Grouping the entries that's referring to the same object can be changeling. Incorrect grouping of different entries will adversely affect the data consistency. This require some level of baby sitting to make sure the program works consistently
- Cross validating data outside the database with maps from 3rd maps and websites, such as Google, Yahoo, Baidu and Wikipedia
    - Potential Issue: requesting data from these companies may not be easy since they are mostly for-profit organizations