

OpenStreetMap Data Case Study

Map Area

Las Vegas, NV, United States

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

<https://www.openstreetmap.org/relation/170117#map=10/36.2670/-115.3510>

I chose Las Vegas as an area of data to analyze since it is a city I am familiar with having visited many times.

Data Audit

Since there were different types of tags being used, I audited and counted the unique number of tags:

def test():

```
tags = count_tags(OSM_FILE)
pprint.pprint(tags)
assert tags == {'bounds': 1,
                'member': 4328,
                'nd': 1181889,
                'node': 994412,
                'osm': 1,
                'relation': 558,
                'tag': 574228,
                'way': 102846}
```

I then looked for patterns in the tags (ex. tags containing problem characters, tags that are only lowercase) using the below regular expression.

```
lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=+/&<>;'\\"?%#$@\\.\. \t\r\n]')
```

Code returned:

```
{'lower': 300441, 'lower_colon': 265414, 'other': 8373, 'problemchars': 0}
```

Problems Encountered in the Map

As suggested, I initially ran code on a small sample size of the area. One of the main problems encountered in the data were the problematic postcodes and different abbreviations for street names.

Postal Codes

Postcode variations:

- 89105-1900
- 8856
- NV 89014
- Nevada 89113
- 6451112

I searched to find whether the postcode matched the criteria of having 5 digits only and if not they were fixed or not included. I did this by running a regular expression for the pattern I was looking for. The code cleans the postcode and returns the 5 desired digits removing leading or trailing characters.

```
def update_postcode(postcode):
    #searches for any pattern that contains 5 consecutive numbers
    #may begin with letters and end with any characters
    # The 5 consecutive numbers are captured:
    search = re.match(r'^\D*(\d{5}).*',postcode)
    # searches if a match is found (ex. a string with 5 consecutive numbers)
    if search:
        clean_postcode = search.group(1)
        #print "clean_postcode"
        #print clean_postcode
        # returns `clean_postcode` and exits function
        return clean_postcode

for postcode in postcodes:
    print update_postcode(postcode)
```

Street Name Abbreviations

A second problem found was inconsistency street name abbreviations. These were some of the variations:

Street abbreviations:

- St. > Street
- Ave > Avenue
- Rd. > Road
- Blvd. > Boulevard
- Pkwy > Parkway
- Cir > Circle
- Dr > Drive

They were corrected using the following mapping:

```
{ "St": "Street",
  "St.": "Street",
  "Ave": "Avenue",
  "Ave.": "Avenue",
  "AVE": "Avenue",
  "Rd.": "Road",
  "blvd": "Boulevard",
  "Blvd": "Boulevard",
  "Blvd.": "Boulevard",
  "Pkwy": "Parkway",
  "Cir": "Circle",
  "Dr": "Drive"
}
```

I used the following code to correct the street names:

```
for street_type, ways in street_types.iteritems():
    for name in ways:
        better_name = update_name(name, mapping)
```

```
print name, "=>", better_name
if name == "S Rainbow Blvd":
    assert better_name == "S Rainbow Boulevard"
```

Size of OSM File

Las-vegas_nevada.osm 213 MB

Database Exploration using SQL

Below I explore the data of the file providing information such as number of nodes, ways, unique users, and who contributed the most. In addition I also run queries on city demographics such as religion and number of bars/pubs.

Number of Nodes

```
SELECT COUNT(*) FROM nodes;
994412
```

Number of Ways

```
SELECT COUNT(*) FROM ways;
102846
```

Number of Unique Users

```
SELECT COUNT(uid) FROM(SELECT uid FROM nodes UNION SELECT uid FROM ways);
1550
```

Top 10 Contributors

```
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|253647
tomthepom|103918
woodpeck_fixbot|73927
alecdhuse|66676
abellao|56785
gMitchellID|46525
robgeb|41212
nmixer|40093
Tom_Holland|34016
MojaveNC|28691
```

Top 5 Religions

```
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 5;
```

```
christian|289
jewish|3
```

bahai|2
muslim|2
buddhist|1

Number of bars/pubs in Vegas

```
SELECT COUNT(*)  
FROM nodes_tags  
WHERE key = 'amenity' and value = 'bar' or value = 'pub';
```

88

Number of bar and pub separate

```
SELECT nodes_tags.value, COUNT(*) as num  
FROM nodes_tags  
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='bar' or value='pub') i  
ON nodes_tags.id=i.id  
WHERE nodes_tags.key='amenity'  
GROUP BY nodes_tags.value;
```

bar	68
pub	20

Additional Improvement Ideas

I found that the way the streets were named were not uniform. For example, South Las Vegas Boulevard was sometimes represented in different ways. I ran this query to show the different ways it was written out. This could be problematic if, for example, you are looking for number of restaurants on South Las Vegas Boulevard.

```
SELECT DISTINCT(value)  
FROM nodes_tags  
WHERE value LIKE "%Las Vegas%" and value NOT LIKE "%North%" and value NOT LIKE "%Parkway%"  
AND key = 'street';
```

*Las Vegas Boulevard S.
South Las Vegas Boulevard
Las Vegas Boulevard S
Las Vegas Blvd South
S Las Vegas Boulevard
Las Vegas Boulevard
Las Vegas Blvd S
Las Vegas Boulevard South
S Las Vegas blvd.*