Data Wrangling Project

By Abir Pattnaik

MAP AREA

Las Vegas,Nevada, United Sites

Download extract from here (https://mapzen.com/data/metro-extracts/metro/las-vegas_nevada/).

Reason for choosing this city:-

I have always been fascinated with this city, the night life the entertainment centres is on my Cities to Visit List.Secondly, the quote "What happens in Vegas Stays in Vegas !" is one of the most popular quotes I have ever heard.Also the uncompressed version of the Las Vegas Extract came out to be 221 MB hence I also ran my auditing codes on it so that I can properly audit the city and do the necessary cleaning.

The process I used:

1. Download the map extract and Unzip It.
2. Check Data for Inconsistencies in Street Names and Postal Codes
3. Updating the Street Names and Postal Codes.
4. Converting those files to csv format after getting cleaned.
5. Use SQL to conduct Queries on the database created from the csv files.

Auditing Data

In the auditing data, I used both the files i.e. 'las-vegas_nevada.osm'(221 MB) and 'las-vegas_nevada_sample.osm'(7.3 MB).The reason I did like this i.e using the whole data and the small data because I wanted to check if my code is working properly or not hence I ran the data with a smaller file.After I was satisfied the code worked properly I used the bigger file and made changes to inconsistencies I faced.

These were the problems I came across :-

a. Abbreviated Street Type: There were street names like "Airport Rd5","5070 Arville St." and many more.

b. Postal Codes:The postal code written are only numbers but I came across 4 types of postal codes. Eg.

```
1. 89148
2. 89104-1307
3. NV 89142
4. Nevada 89113
```

and some wrong postal codes.

c. Another problem I came across was this.'#' and 'numbers' at the end of street names.e.g. "Fremont Street Ste. 1170","E SILVERADO RANCH BLVD Suite 120" , "E Tropicana Ave #A".These were difficult to edit.

Street Types Auditing

In detecting of abbreviated street names I used regular expressions to find the end street types.They were solved using mapping.I.e. For e.g. mapping={"St":"Street", "Cir":"Circle","ln":"Lane"}

```
def update_name(name,mapping):
    m=street_type_re.search(name)
    street_type=m.group()
    if m:
        street_type = m.group()
        if street_type not in expected:
            if street_type in mapping.keys():
                name = re.sub(street_type_re, mapping[street_type], name)
    return name
```

The abbreviated word simply substitutes with the word that I provided as suitable in the mapping dictionary.

Result

```
5070 Arville St. => 5070 Arville Street

South Pecos Road Suite 103 => South Pecos Road Suite 103

Paradise Rd => Paradise Road

Losee Rd => Losee Road

El Camino Rd => El Camino Road

W Craig Rd => W Craig Road

Las Vegas Boulevard S. => Las Vegas Boulevard South

W Sahara ave => W Sahara Avenue

N Rainbow blvd => N Rainbow Boulevard
```

Postal Codes Auditing

While assessing the osm data I came across 4 types of Post Codes.

1.89052

2.89104-1307 (http://usplaces.com/Hotels%20and%20Motels/las-vegas-nv/super-8-hotel.5)

3.NV 89119 => 89119

4.Nevada 89113 => 89113

1 and 2 are correct postcode notations.2 is also correct as I searched in the internet and **they** were location of that code.3 and 4 needed to be changed. Hence,I used regular expressions for them.Explanation of each regular expression I have is given in comments in my **Audit_Section.py** file

```
post_code_re=re.compile(r'(^(89)\d{3}$)|(^(89)\d{3}-\d{4}$)')
post_code_re_NV=re.compile(r'^(NV)\s(89)\d{3}$')
post_code_re_Nevada=re.compile(r'(Nevada)\s(89)\d{3}')
def update_postcode(postcode):
    m=post_code_re.search(postcode)
    NV_abbr=post_code_re_NV.search(postcode)
    NV_full=post_code_re_Nevada.search(postcode)
    if m:
        return m.group()
    elif NV_abbr:
        NV_abbr_group=NV_abbr.group()
        # print NV_abbr_group
        return NV_abbr_group[3:]
    elif NV_full:
        NV_full_group=NV_full.group()
        # print NV_full_group
        return NV_full_group[7:]
    else:
        return "NOT CORRECT POSTCODE"
```

Result

```
89052 => 89052

89147-8491 => 89147-8491

89148 => 89148

89191 => 89191

89104-1307 => 89104-1307

89032 => 89032

NV 89119 => 89119

891171 => NOT CORRECT POSTCODE

Nevada 89113
```

Preparing for database

The **Database.py** file used the Audit_section.py and converted into 5 csv files.The las-vegas_nevada_sample.osm was used for conversion.

File Sizes

las-vegas_nevada.osm ....221.04 MB

las-vegas_nevada_sample.osm............11.24 MB

las-vegas_nevada.db...................145.86 MB nodes.csv.............................86.43 MB

nodes_tags.csv.........................2.39 MB

ways.csv...............................6.68 MB

ways_nodes.csv........................30.56 MB

ways_tags.csv..........................14.85 MB

Number of Nodes

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

```
1047575
```

Number of Ways

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

```
113077
```

Number of unique users

```
SELECT COUNT(DISTINCT(x.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) x
```

```
1117
```

Top 5 contributing users

```
SELECT x.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) x
GROUP BY x.user
ORDER BY num DESC
LIMIT 5
```

```
alimamo "251405"

tomthepom "121115"

woodpeck_fixbot "70515"

alecdhuse "66521"

abellao "55537"
```

New explorations

Top 5 cities in the data

```
SELECT value , COUNT(*)
FROM nodes_tags WHERE key ='city'
GROUP BY value
ORDER BY count(*)
DESC limit 5;
```

```
Las Vegas "315"

Henderson "31"

North Las Vegas "10"

Boulder City "4"

Boulder City NV "2"
```

Casino Count

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

```
casino 10
```

List of casinos

```
SELECT nodes_tags.value
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='casino') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'
GROUP BY nodes_tags.value
```

```
Binion's Gambling Hall and Hotel

Dotty's

Dotty's casino

Encore Casino at Wynn

Gold Strike

Jerry's Nugget Casino

Jokers Wild Casino

Poker Palace

Wynn Las Vegas Casino
```

Top 5 amenities in the area

```
SELECT VALUE , COUNT(*)
FROM nodes_tags WHERE key = 'amenity'
GROUP BY VALUE
ORDER BY COUNT(*)
DESC LIMIT 5;
```

```
restaurant "478"

place_of_worship "294"

fuel "284"

fast_food "280"

fountain "266"
```

Top 5 shops available in the area

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

> convenience "180"
>
> clothes "61"
>
> supermarket "60"
>
> car_repair "40"
>
> mobile_phone" "33"

Top 5 popular cuisines

```
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 5;
```

> mexican "41"
>
> pizza "31"
>
> american "20"
>
> italian "18"
>
> steak_house "16"

Inferences achieved from the queries

1. Alimamo is the highest contributor.
2. Main data was collected from Las Vegas area and then Henderson.
3. Casino that were available in the area was 10.
4. Restaurants are the highest in the area followed by place of worship.
5. As understood,convenience stores are the highest in the area.
6. I suppose Mexican cuisine is quite famous in Las Vegas.

CONCLUSION

The Las Vegas XMl file was used for auditing and I only audited street names and postcodes.Postcodes were removed easily and all of the postcodes were cleaned properly.

However, in the case of street Names I came across street names ending with '#' or numerics.These led to street names like "S. Valley View Blvd. Ste 10", "Polaris Ave #16","E Sahara Blvd #15".

Also while doing queries,due to improper naming I couldn't properly count the total no. of casinos.For e.g. there is Dotty and there is Dotty's casino.

This is because I used DB browser for SQLite which helped me to examine data properly.Apart from street names and postcode many other are uncleaned hence need to be cleaned.This can be due to improper human inputs.

But overall there were less problematic characters to clean.

Limitations

One of the major limitations faced is that OpenStreetMap is opensource that leads to human error formats that may or may not be correct hence creates problem for users.

Suggestions

1.One suggestion I can think of is using pre-known words to define a street name.This would be a boon for normal users who are editing the OSM.For eg. Due to speed typing humans write AVE. instead of Avenue.But if OSM makes interactive by suggesting the user to put Avenue it would be quite faster.

2.Another one I can think is frequent validation.Every Week or Every 3-4 days frequent checks on new updates can be checked if they have the correct notation.That way whoever uses gets a cleaned file always.

Benefits and Anticipated problems

BENEFITS: One of the benefits on implementing the suggestions is that a cleaned data can be used by both users and companies.I came across such company where they used OSM.So, if a cleaned OSM file is there it is much easier to work upon and analyis would be much better to work upon like where to install new clothing line stores,casinos where it is easier to reach and many more ideas.

PROBLEM: Frequent validation requires experienced professionals and need a lot of time to validate hence time and money is a problem.Open Street Map quite relies on normal users hence may go down as repeated validation and just selecting from pre-known words **might** be a problem.

References

1. https://www.w3schools.com/xml/xpath_syntax.asp (https://www.w3schools.com/xml/xpath_syntax.asp)
2. https://docs.python.org/2/library/xml.etree.elementtree.html#module-xml.etree.ElementTree (https://docs.python.org/2/library/xml.etree.elementtree.html#module-xml.etree.ElementTree)
3. https://stackoverflow.com/questions/45771809/how-to-extract-and-visualize-data-from-osm-file-in-python (https://stackoverflow.com/questions/45771809/how-to-extract-and-visualize-data-from-osm-file-in-python)
4. https://discussions.udacity.com/t/lesson-13-quiz-10/315576/7?u=abir.pattanaik (https://discussions.udacity.com/t/lesson-13-quiz-10/315576/7?u=abir.pattanaik)
5. https://discussions.udacity.com/t/p3-openstreetmap---cleaning-street-names/238014?u=abir.pattanaik (https://discussions.udacity.com/t/p3-openstreetmap---cleaning-street-names/238014?u=abir.pattanaik)
6. http://effbot.org/zone/element-iterparse.htm (http://effbot.org/zone/element-iterparse.htm)
7. https://developers.google.com/edu/python/regular-expressions (https://developers.google.com/edu/python/regular-expressions)
8. https://discussions.udacity.com/t/p3-project-combining-auditing-cleaning-and-csv-creation/231037/2 (https://discussions.udacity.com/t/p3-project-combining-auditing-cleaning-and-csv-creation/231037/2)
9. http://www.geonames.org/postal-codes/US/NV/nevada.html (http://www.geonames.org/postal-codes/US/NV/nevada.html)
10. http://sqlitebrowser.org/ (http://sqlitebrowser.org/)
11. https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)