OpenStreetMap Case Study

Cassandra Lamendola

Portland, Oregon, U.S.

- openstreetmap.org
- mapzen.com

Portland, Oregon is one of my favorite cities. I visit there at least once a year because I love the culture of the city. That's why I chose to use the Portland OpenStreetMap data for this case study.

Problems Encountered in the Map

- Abbreviated street types ('Ave.', 'St.', 'Blvd.', etc.)
- Abbreviated street directions (S, NE, W)
- Inconsistency in zip codes (some include 4 digit secondary zip code, some zip codes are outside of Portland)
- Inconsistent or upper/lower case city names ('molalla', 'Portland, OR', 'WOODBURN')

Some Solutions

Abbreviated Street Types

To correct the abbreviated street types, I used the regular expression from an example in the SQL lessons. This is the function from my PortlandCaseStudy.py file that I wrote to do the correction:

```
def update_name(name, mapping):
    name = name.split(" ")
    old_street_type = name.pop()
    if old_street_type in mapping:
        name.append(mapping[old_street_type])
    else:
        name.append(old_street_type)
    name = " ".join(name)
    return name
```

So the address "SW Martinazzi Ave" becomes "SW Martinazzi Avenue".

Abbreviated Street Directions

In order to correct the abbreviated street directions, I decided to update the function I had written for correcting street types so that it could take care of both jobs. I also updated the mapping dictionary to include possible directions. Here is the updated function from PortlandCaseStudy.py:

```
def update_name(name, mapping):
    name = name.split(" ")
    for word in range(len(name)):
        if name[word] in mapping.keys():
            name[word] = mapping[name[word]]
    name = " ".join(name)
    return name
```

The same address from above, "SW Martinazzi Ave" is changed to "Southwest Martinazzi Avenue".

Zip Codes

To check for problematic zip codes, I used the following function (Zip codes in the city of Portland begin with 972):

```
def audit_zip_codes(zip_code):
    if zip_code[:3] != "972" or len(zip_code) != 5:
       zip_codes.append(zip_code)
```

I was surprised to find that there were no problems with incorrect characters or improper lengths of zip codes (lengths other than 5 or 9). The only issues I found were:

1. Several zip codes that did not begin with 972
2. Inconsistency with some zip codes containing additional 4 digits

After a quick google search of some of those suspicious zip codes, I found that they were all, in fact, from cities surrounding Portland. When I thought about fixing the problem with length inconsistency, I realized there was really no good solution. I didn't know what the additional 4 digits were for, but after research I found out that they actually designate a specific delivery route for USPS. It is not helpful to remove the last 4 digits of 9 digit codes because that would in fact make them less accurate and there is no practical way to update all 5 digit codes. Some addresses don't even have the additional 4 digits. So this field is actually perfectly fine!

Inconsistent City Names

I wrote the following function to normalize city names:

```
def update_city(city):
    if city[0].islower():
        city[0].upper()
    if city.isupper():
```

```
        city.lower()
        city[0].upper()
    if city.find(',') != -1:
        city.split(',')
        city = city[0]
    return city
```

After the update, 'WOODBURN' became 'Woodburn' and 'Portland, OR' became 'Portland'

Overview of the Data

Number of Unique Users

```
SELECT COUNT(DISTINCT(subq.uid))
FROM (
    SELECT uid
    FROM nodes
    UNION
    SELECT uid
    FROM ways)
AS subq;
```

```
1169
```

Number of Nodes

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

```
6468932
```

Number of Ways

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

```
831726
```

Number of Cafes

```
SELECT COUNT(*)
FROM nodes_tags
WHERE value = 'cafe'
OR value = 'coffee_shop';
```

```
556
```

Number of Bus Stops

```
SELECT COUNT(*)
FROM nodes_tags
WHERE value = 'bus_stop';
```

```
3009
```

Top 5 Amenities in Portland

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

```
bicycle_parking,2453
place_of_worship,882
bench,822
waste_basket,756
restaurant,653
```

Other Ideas About the Database

Improving the Database Further

One way the data could be improved would be to have more consistency in naming amenities, cities, etc. I noticed, for example, that 'cafe' and 'coffee_shop' were both used. I'm sure this problem occurred in other categories as well.

This could be fixed by implementing data validation, so that users can only enter data that meets certain validation rules. For example, an error would be thrown if a user tried to enter a street name in all caps.

This solution has some limitations when applied to ammenities, because there may be a type of amenity that was not previously considered when creating the validation rules.

Number of Streets with a Dedicated Bike Path

I've heard that Portland is a very bicycle-friendly city. This makes me wonder how many streets in Portland have a dedicated bike path.

```
SELECT COUNT(*)
FROM (
    SELECT *
    FROM (ways_tags JOIN ways
        ON ways_tags.id = ways.id)
    WHERE ways_tags.key = 'bicycle'
    AND ways_tags.value = 'yes'
    OR ways_tags.value = 'designated');
```

```
39323
```

What percentage of non-highway streets have bike paths?

```
SELECT COUNT(DISTINCT(subq.id))
FROM (
    SELECT *
    FROM ways JOIN ways_tags
    ON ways.id = ways_tags.id
    WHERE ways_tags.key != 'highway')
AS subq;
```

```
807563
```

That means there are 39,323 ways containting bike paths out of 807,563 total ways. So almost 5% of streets in Portland have a designated bike path. I would like to see how this compares with other cities in the US.