Wrangle OpenStreetMap Data

Sai Venkat Kotha

Map Area

San Francisco, CA, USA

Problems encountered in the map

- Abbreviated street names and multiple abbreviations for the same street type
- Wrong post codes
- Improper house numbers

Street Names

The osm file contained street names which were abbreviated and there was inconsistency in the abbreviations i.e street names of the same kind had different abbreviations. To clean the street names the approach used was to remove the abbreviations and to use the complete street names to ensure consistency.

The auditing phase revealed all the street names that had this problem i.e abbreviated street names. The results of the auditing was used to create a mapping which maps the abbreviated street names to the respective complete name. This mapping is used to transform an abbreviated street name to its respective complete name.

```
def clean_street(element, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            if street_type in street_mapping:
                new_street_type = street_mapping[street_type]
                new_street_name = street_name[:len(street_name)-len(street_type)]+new_street_type
                return new_street_name
            else:
                return street_name
        else:
            return street_name
    else:
        return street_name
```

The above code snippet shows the function used to clean the street names. The expected street types are stored in the list "expected" and the mapping is stored in the dictionary "street_mapping". A regular expression is used to extract the street type from the street name. If a street name has a street type that is not present in the expected list, the street mapping dictionary is used to transform the street name to a new street name with the complete street type.

Post Codes

All the post codes of San Francisco should contain five digits and should start with the digit '9'. The osm file had many post codes which did not have this form. The auditing phase revealed such wrong post codes.

Some of the post codes in the osm file with the wrong format are '*94103-3124*', '*CA 94560*', '*CA*' etc.

Such wrongly formatted post codes are cleaned to maintain uniformity. The cleaning approach used was to check if the post code had a string of five digits that started with the digit '9' and if such string is found, the bad post code is replaced with this string. If the post code did not contain such kind of string the value "unknown" is used as post code because any other value would be wrong and would not represent the post code properly.

```
def clean_postcode(element, postalcode):
    postcode_re = re.compile(r'9[0-9]{4}')
    if ((len(postalcode) != 5) or (postalcode[0] != '9')):
        m = postcode_re.search(postalcode)
        if m:
            postcode = m.group()
        else:
            postcode = "unknown"
    else:
        postcode = postalcode
    return postcode
```

The above code snippet shows the function that cleans the post codes. The value of the post code is checked to determine if it has five digits and starts with the digit '9'. If this is not the case, a regular expression is used to find the required pattern of post code and if such pattern exists, the bad post code is replaced with this pattern.

House Numbers

A house number is expected to contain either numerical characters or alpha-numeric characters. While auditing the house numbers it was found that the osm file contained many house numbers which did not obey this format.

Some of the improper house numbers are '*2218+2216*', '*915-*' etc. Some of the house numbers had multiple fields or multiple house numbers. Such values of house numbers were not changed to a single value because there is no proper logic to find out what values to eliminate or what value to keep.

The osm file also had house numbers which included a '*Suite number*', but there are instances where the word '*Suite*' is abbreviated as '*ste*' or '*ste.*' etc.

```
def clean_housenumber(element):
    house_number = element.get('v')
    if re.match(hnumber_re, house_number) is None:
        if house_number[len(house_number)-1] == "-":
            new_house_number = house_number.replace("-","")
        elif ";" in house_number:
            new_house_number = house_number.replace(";",",")
        elif re.match(bad_hnum_re, house_number) is not None:
            if "-" in house_number:
                new_house_number = house_number.replace("-","")
            else:
                new_house_number = house_number.replace(" ","")
        elif re.search(hnum_suite_re, house_number) is not None:
            m = hnum_suite_re.search(house_number)
            pattern = m.group()
            index = house_number.find(pattern)
            if index == 0:
                new_house_number = "Suite"+house_number[len(pattern):]
            else:
                new_house_number = house_number[:index]+"Suite"+house_number[index+len(pattern):]
        elif re.match(hnum_plus_re, house_number) is not None:
            new_house_number = house_number.replace("+","-")
        else:
            new_house_number = house_number
    else:
        new_house_number = house_number
    return new_house_number
```

The above code snippet shows the function that cleans the house numbers. A house number which is in improper format could be in multiple formats. Some house numbers had '-' at the end, such house numbers were cleaned by removing the '-'. In all the house numbers which had multiple values, in some house numbers the values were separated using ';'(semi-colon) and in some house numbers ','(comma) was used to separate the values. To ensure uniformity only ','(comma) is used i.e in the house numbers which had ';', ';' is replaced with ','.

The house numbers in which the word '*Suite*' is abbreviated, the abbreviation is replaced with the complete word '*Suite*'. The house numbers which had a '+'(plus) sign, the '+' sign is replaced with '-' in such house numbers. There are many house numbers which contain alpha-numeric characters in the form '221B'. But there is no consistency to this format i.e there are instances where the house numbers are '221 B' or '221-B'. This issue is resolved by transforming such house numbers to the format ‘221B'.

Overview of the data

File Sizes

nodes.csv - 552 MB
nodes_tags.csv - 9.5 MB
ways.csv - 49.8 MB
ways_nodes.csv - 187.7 MB
ways_tags.csv - 58.5 MB

Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
6582698
```

Number of nodes with tags

```
sqlite> SELECT COUNT(DISTINCT id) FROM nodes_tags;
95980
```

Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
814583
```

Number of ways with tags

```
sqlite> SELECT COUNT(DISTINCT id) FROM ways_tags;
807653
```

Number of unique users

```
sqlite> SELECT COUNT(uid)
FROM (SELECT uid FROM nodes UNION SELECT uid FROM ways);
2717
```

Number of schools in San Francisco

```
sqlite> SELECT COUNT(*)
FROM (SELECT value FROM nodes_tags WHERE value = "school" UNION
ALL SELECT value FROM ways_tags WHERE value = “school");
2592
```

Number of restaurants in San Francisco

```
sqlite> SELECT COUNT(*)
FROM (SELECT value FROM nodes_tags WHERE value = "restaurant" UNION
ALL SELECT value FROM ways_tags WHERE value = “restaurant");
6319
```

Top 10 users

```
sqlite> SELECT user, COUNT(*)
FROM (SELECT uid, user FROM nodes UNION ALL SELECT uid, user FROM
ways)
GROUP BY uid
ORDER BY COUNT(*) DESC
LIMIT 10;
```

user	count
andygol	1496880
ediyes	887946
Luis36995	668832
dannykath	546062
RichRico	411052
Rub21	383453
calfarome	186650
oldtopos	166208
KindredCoda	148684
karitotp	138915

Top 10 amenities

```
sqlite> SELECT value, COUNT(*)
FROM (SELECT value FROM nodes_tags WHERE key = "amenity" UNION ALL
SELECT value FROM ways_tags WHERE key = “amenity")
GROUP BY value
ORDER BY COUNT(*) DESC
LIMIT 10;
```

amenity	count
restaurant	6314
parking	4853
bench	2403
cafe	2067
school	1884
place_of_worship	1856
post_box	1374
fast_food	1296
bicycle_parking	1143
drinking_water	1056

Additional improvements

```
sqlite> SELECT DISTINCT value
FROM (SELECT value FROM nodes_tags UNION ALL SELECT value FROM
ways_tags)
WHERE value LIKE ‘%parking%'
LIMIT 20;
```

```
parking
parking_entrance
bicycle_parking
UCB Parking
at&t ballpark valet bycycle parking
Public Parking
Alameda County Parking Garage Heliport
Priority Parking
California Parking
Lake Merritt Bike Link Parking
Children's Hospital & Research Center Oakland - Parking Garage
La Costanera Parking
Post Office Parking
Kaiser Center Parking Garage
City Center Parking Garage
Public Parking Garage
Museum Parking Garage
parking_aisle
Peralta MOB Parking Garage
Walgreens & 24 Hour Fitness Underground Parking Garage
```

The results of the above query indicate that there is no consistency to the values of the tags. The values 'parking', 'Public Parking', 'California Parking', 'Public Parking Garage' represent a parking space but they are all considered as different amenities because of their different values. This problem can be solved by giving a same value to all the amenities that represent a parking space. But this could also lead to another problem where the amenities are all generalized and the tags cannot provide specific details about them.