P3: Wrangling a data set from Open Street Map and creating a SQL database from it

Summary

This is the final project of Data Wrangling course in Udacity Data Analytics Nanodegree. The objective of this project is wrangling a data set related to an arbitrary area in Open Street map in XML format and structure the data for creating a SQL database. After putting data in the database, we want to look at the data and provide some insights from it.

Map Area

Santa Monica, California, United States

MapZen: https://mapzen.com/data/metro-extracts/your-extracts/15217bd59e1c

OpenStreetMap: https://www.openstreetmap.org/relation/3353288#map=11/33.9835/-118.4010

This map is related to Santa Monica area in Los Angeles County, California. I like this region and I am passionate about learning more details about it.

Approach

For having an idea about the map and which kind of problems it has, I use an iterative approach. First, I create the CSV files from the OSM file (the original XML file). Then I used sqlite data base and queried the data with different questions and when I got an idea about the problems, I started over and defined different functions for catching problems and different update functions for fixing them. Finally, I used those update functions for cleaning the problematic parts and create final CSV files from the XML file and create the SQL database in sqlite environment and made some queries base on the cleaned data for understanding more details about the selected region.

Problems encountered in the map

I found these problems in the data set:

- Problematic characters in any field
- Nodes that contain tag as key = ‘type’
- Street part Problems
    - Abbreviation in Street names.
    - Abbreviation in Street prefixes.
    - Lack of any kind of Street at the end of some street address
    - Street name starting with Lowercase letter
    - House number included in Street name
    - Suite word and its value included in Street name
- House Number Problems
    - The whole address of the node included in the house number
    - Suite word and its value included in house number

- State abbreviation Problems
    - Different type of abbreviating State
- Post Code Problems
    - State acronym included in the Post Code
- Phone Number Problems
    - Wrong format
    - Different format for phone numbers
- Website address Problems
    - Different keys for storing website addresses
    - Problematic Website addresses

Problematic characters in any field

If any value in any field has problematic characters, I put aside this like in XML file and nothing will be transferred in the CSV file related to this line.

Nodes that contain tag as key = 'type'

There are some nodes that contain a tag as key = 'type' and we have node.type as part of our schema(as last column in the node table); I only mention this because I creates validity problem.

Street part Problems

There are different types of problems in street part of address:

- Abbreviation in Street names
  There are many types of streets in this data set; Some of them are abbreviated; like: St., Bvd, Ave, Dr. For solving this problem, I defined a dictionary for mapping the abbreviated word to the completed one.
- Abbreviation in Street prefixes
  Some street names have abbreviated prefixes. e.g. E, W, S, S., N.
  For solving this problem, I provide another dictionary that contains different kinds of abbreviation as keys and complete prefix as values. e.g. S. or S will map to South.
- Lack of any kind of Street at the end of some street address
  Some street names have no street type at all. e.g. Pico is a street name and it needs Boulevard as street type. So, I put this kind of problems in the mapping dictionary and add appropriate street type to it. "Pico": "Pico Boulevard".
- Street name starting with Lowercase letter
  For solving this type of problems, I used *string* library and *capwords*() method.
  e.g. "veteran avenue" will modify to "Veteran Avenue".
- House number included in Street name
  Some street names include house number. For uniformity, I separate these numbers and create a new line for them in the CSV file.
  e.g. "1200 South Sepulveda" will transform to two different records:
  housenumber: 1200; street: South Sepulveda; The new line will inherit "id" and any other fields from the original one.
- Suite word and its value included in Street name

Some street addresses contain Suite part and its value. For uniformity, I put Suite number in a new line with its value. e.g. '15th Street Ste. 1101'.

Here is the update function for street part:

```
def update_name(name, mapping_street, mapping_abbrev):
    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        #If the name exist in mapping keys, it means that's a problem and we should fix it.
        if street_type in mapping_street.keys():
            name = re.sub(street_type, mapping_street[street_type], name)
    # Updating W , E, N, S to West, East, North, South; if they are at the begining of an address.
    m_1 = street_abbrev_re.search(name)
    if m_1:
        street_abbrev = m_1.group()
        if street_abbrev in mapping_abbrev.keys():
            name = re.sub(street_abbrev, mapping_abbrev[street_abbrev], name)
    # capitalizing first letter of all words in problematic address
    name = string.capwords(name)
    return name
```

And here is the shape_element function part, related to fixing street problems:

```
# Fixing Street names
if is_street_name(el_tag):

    # updating problematic words with defined maps
    el_tag.attrib['v'] = update_name(el_tag.attrib['v'], mapping_street, mapping_abbrev)

    # adding a new line for housenumber, if it's mentioned in street address
    if housenumber_in_street_re.search(el_tag.attrib['v']):
        value_list_1 = el_tag.attrib['v'].split(' ')
        # keeping housenumber for creating a new line and deleting it from street address
        housenumber = value_list_1.pop(0)
        el_tag.attrib['v'] = ' '.join(value_list_1)
        add_new_line('housenumber', housenumber, tags, element)

    # adding a new line for suite, if it's mentioned in street address
    if (ste_re.search(el_tag.attrib['v'])) or (suite_re.search(el_tag.attrib['v'])):
        value_list_2 = el_tag.attrib['v'].split(' ')
        # keeping suite number for creating  a new line
        suite_number = value_list_2.pop(-1)
        # removing the suite word from the address
        value_list_2.remove(value_list_2[-2])
        el_tag.attrib['v'] = ' '.join(value_list_2)
        # adding suite line
        add_new_line('Suite', suite_number, tags, element)
```

House Number Problems

I found these two kinds of problems in this part:

- The whole address of the node included in the house number
  Some house number values are contained the whole address. I separate each part and create a new line for each of them.
  e.g. "1850 Sawtelle Boulevard, Suite 300, Los Angeles, CA 90025" will transfer to 6 new lines.
- Suite word and its value included in house number
  It's not supposed to mention suit number in house number. So, again I separate this part and create a new line for it.
  e.g. "12301 Suite 650" will transfer to two different parts: housenumbers and suit part.

Here is the shape_element function part, related to fixing housenumber problems:

```
# Fixing House Numbers problems
if is_housenumber(el_tag):
    # fixing the only housenumber that contains the whole address
    if el_tag.attrib['v'].startswith('1850 Sawtelle Boulevard'):
        value_list_3 = el_tag.attrib['v'].split(', ')
        el_tag.attrib['v'] = value_list_3[0].split(' ')[0]
        # adding suite line
        add_new_line('Suite', value_list_3[1].split(' ')[-1], tags, element)

        # adding city line
        add_new_line('city', value_list_3[2], tags, element)

        # adding state line
        add_new_line('state', value_list_3[3].split(' ')[0], tags, element)


    # adding a new line for suite, if it's mentioned in housenumber
    elif (ste_re.search(el_tag.attrib['v'])) or (suite_re.search(el_tag.attrib['v'])):
        value_list_2 = el_tag.attrib['v'].split(' ')
        # keeping suite number for creating  a new line
        suite_number = value_list_2[-1]
        el_tag.attrib['v'] = value_list_2[0]
        # adding new line for suite
        add_new_line('Suite', suite_number, tags, element)
```

And here is the add new line function:

```
def add_new_line(key, value, tags, element, types = 'addr'):
    temp= {}
    temp['id'] = element.attrib['id']
    temp['key'] = key
    temp['type'] = types
    # using Uppercase for first letter of all words
    temp['value'] = string.capwords(value)
    tags.append(temp)
```

State abbreviation Problems

- Different type of abbreviating State
  This is the only problem I found in state field. By different type I mean this:
  "CA", "Ca", "CA,"
  For unfirming these values, I choose all of them to be like the first one: "CA"
  For being efficient I only check the state and if there is a problem I change it to "CA".

```
def update_state(state, state_list):
    if state in state_list:
        state = "CA"
    return state
```

Post Code Problems

- State acronym included in the Post Code
  This is the only problem I found in PostCode field.
  e.g. "CA 90272" will transfer to two lines. One line is State and the other is post code.

```
# Fixing Post Codes
if is_postcode(el_tag):
    # finding problematic postcodes that contain state as part of postcode
    if state_in_postcode_re.search(el_tag.attrib['v']):
        value_list_4 = el_tag.attrib['v'].split(' ')
        el_tag.attrib['v'] = value_list_4[1]
        # adding new line for state
        add_new_line('state', value_list_4[0], tags, element)
```

Phone Number Problems

- Wrong format
  Some phone numbers are not in a valid format.
  e.g. "01-310-260-6308" will change to "+1-310-260-6308".
- Different format for phone numbers
  There are many different formats for phone numbers in this file. For putting them in one format, I used *ponenumbers* library and *format_number* method.

Here is the shape_element function part, related to fixing phone numbers problems:

```
# Fixing Phone numbers and format them
if is_phone(el_tag):
    if el_tag.attrib['v'] in notValidPhones:
        el_tag.attrib['v'] = el_tag.attrib['v'].replace('0', '+', 1)
    # Uniform all phones the same like this: +13102606308
    if el_tag.attrib['v'].startswith("+"):
        el_tag.attrib['v'] = el_tag.attrib['v'][1:]
    z = phonenumbers.parse(el_tag.attrib['v'], "US")
    el_tag.attrib['v'] = phonenumbers.format_number(z, phonenumbers.PhoneNumberFormat.E164)
```

Website address Problems

- Different keys for storing website addresses
  There are two different keys for storing website addresses: “url” and “website”
  Most of the addresses are mentioned with “website” key and there are few ones with “url” key. So, I decided to change the “url” key to “website” for uniformity.

- Problematic Website addresses
  There are some website addreses that doesn’t work. For catching this kind of website addresses I used validators library and url() method.
  e.g. this website is not a valid website address:

```
'http://www.dot.ca.gov/hq/tsip/gis/datalibrary/gisdatalibrary.html, bing'
```

I defined this function for finding this kind of problems:

```
def audit_website(problemWebsite, website):
    if not website.startswith('http'):
        website = 'http://' + website
    if not validators.url(website):
        problemWebsite.append(website)
```

For fixing these two problems I put this part in shape_element function:

```
# changing key = 'url' to key = 'website'; because there are many \
# website addresses and only 4 urls as keys;
if key_tag == 'url':
    key_tag = 'website'

# fixing problematic websites
if key_tag == 'source':
    if el_tag.attrib['v'] in problemWebsite:
        el_tag.attrib['v'] = el_tag.attrib['v'].split(',')[0]
```

Now all problems I found are modified and we can go forward to the next step. I mean creating CSV files and creating the Sqlite database and find some specification about the area.

Overview of the Data

This section contains some numeric points about the dataset analyzed.

File Sizes

```
santaMonica.osm ..... 143.8 MB
openStreet.db ..... 88.3 MB
nodes.csv ..... 58 MB
nodes_tags.csv ..... 0.245 MB
ways.csv ..... 4.3 MB
ways_nodes.csv ..... 17.5 MB
ways_tags.csv ..... 14.2 MB
```

Number of Different Tags

member 4634
nd 710878
node 638960
relation 1031
tag 436054
way 63953

Number of unique contributors

350

Different Tag Patterns

Lower Case Tags 249540
Lower with Colon Tags 184694
Other Tags 1819
Problematic Tags 1

Top Contributor Users

```
c.execute("SELECT user, COUNT(*) as count\
           FROM nodes\
           GROUP BY user\
           ORDER BY count DESC\
           LIMIT 10;")

rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'kingrollo', 90092),
 (u'ridixcr_import', 73949),
 (u'Luis36995_labuildings', 71663),
 (u'manings_labuildings', 58496),
 (u'calfarome_labuilding', 56562),
 (u'schleuss_imports', 47011),
 (u'dannykath_labuildings', 42840),
 (u'Jothirnadh_labuildings', 35713),
 (u'saikabhi_LA_imports', 31338),
 (u'kingrollo_imports', 22558)]
```

Number of Users Contribute Only Once

```
c.execute("SELECT user, COUNT(user) as count\
           FROM nodes\
           GROUP BY user \
           HAVING count = 1;")

rows = c.fetchall()
print "Number of one time Contributers: ", len(rows)
```

```
Number of one time Contributers:  85
```

Other ideas about the datasets

In this part I am looking at data and answer some questions with Sqlite data base which I created base on the data I have;

Cities exist in Santa Monica Region

```
city_query = """SELECT tags.value, COUNT(*) as count\
                FROM (SELECT * FROM nodes_tags UNION ALL\
                SELECT * FROM ways_tags) tags\
                WHERE tags.key = 'city'\
                GROUP BY tags.value\
                ORDER BY count DESC;"""
c.execute(city_query)
rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'Santa Monica', 142),
 (u'Los Angeles', 42),
 (u'Venice', 34),
 (u'Marina Del Rey', 6),
 (u'Los Angeles-venice', 3),
 (u'West Los Angeles', 2),
 (u'Pacific Palisades', 1),
 (u'Venice Ca', 1)]
```

Which kind of amenities are in Santa Monica Region?

```
c.execute("SELECT tags.value, count(*) as count\
           FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) as tags \
           WHERE tags.key = 'amenity' \
           GROUP BY value \
           ORDER BY count DESC \
           LIMIT 10;")
rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'parking', 178),
 (u'restaurant', 90),
 (u'bicycle_rental', 81),
 (u'school', 55),
 (u'place_of_worship', 52),
 (u'cafe', 45),
 (u'fast_food', 24),
 (u'drinking_water', 23),
 (u'fuel', 22),
 (u'hospital', 17)]
```

Which restaurant Cuisines are popular in this region?

```
c.execute("SELECT nodes_tags.value, COUNT(*) as count \
           FROM (SELECT tags.id FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) \
           as tags \
           WHERE tags.value = 'restaurant') as restaurant_tags, nodes_tags\
           WHERE restaurant_tags.id = nodes_tags.id AND nodes_tags.key = 'cuisine'\
           GROUP BY nodes_tags.value\
           ORDER BY count DESC \
           LIMIT 10;")
rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'american', 8),
 (u'italian', 5),
 (u'mexican', 3),
 (u'regional', 3),
 (u'burger', 2),
 (u'indian', 2),
 (u'sandwich', 2),
 (u'argentinian', 1),
 (u'asian', 1),
 (u'chinese', 1)]
```

Which religions are related to Worship Places?

```
c.execute("SELECT nodes_tags.value, COUNT(*) as count\
           FROM (SELECT tags.id FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) as tags\
           WHERE tags.value = 'place_of_worship') as worship_place, nodes_tags\
           WHERE worship_place.id = nodes_tags.id AND nodes_tags.key = 'religion'\
           GROUP BY nodes_tags.value\
           ORDER BY count DESC;")

rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'christian', 45), (u'buddhist', 2)]
```

Denomination of Worship Places

```
c.execute("SELECT nodes_tags.value, COUNT(*) as count\
           FROM (SELECT tags.id FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) as tags\
           WHERE tags.value = 'place_of_worship') as worship_place, nodes_tags\
           WHERE worship_place.id = nodes_tags.id AND nodes_tags.key = 'denomination'\
           GROUP BY nodes_tags.value\
           ORDER BY count DESC;")

rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'baptist', 11),
 (u'methodist', 4),
 (u'catholic', 3),
 (u'lutheran', 3),
 (u'mormon', 2),
 (u'nichiren', 1),
 (u'presbyterian', 1)]
```

Coffee Shop brands in Santa Monica

```
c.execute("SELECT nodes_tags.value, COUNT(*) as count \
           FROM (SELECT tags.id FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) as tags\
           WHERE tags.value = 'cafe') as cafe, nodes_tags\
           WHERE cafe.id = nodes_tags.id AND nodes_tags.key = 'name'\
           GROUP BY nodes_tags.value\
           ORDER BY count DESC\
           LIMIT 5;")

rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'Starbucks', 5),
 (u'Coffee Bean & Tea Leaf', 2),
 (u'Starbucks Coffee', 2),
 (u'Blue Bottle Venice', 1),
 (u'Bondi Harvest', 1)]
```

Most common Zip Codes

```
c.execute("SELECT tags.value, COUNT(*) as count \
           FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) as tags\
           WHERE tags.key = 'postcode'\
           GROUP BY tags.value\
           ORDER BY count DESC\
           LIMIT 5;")

rows = c.fetchall()
pprint.pprint(rows)
```

```
[(u'90025', 186),
 (u'90291', 45),
 (u'90405', 45),
 (u'90404', 34),
 (u'90401', 33)]
```

Conclusion and Suggestions

One quality problem I found in this dataset is about *completeness*.
For example, there are more than 100 restaurants or more than 60 health related places registered in Santa Monica Chamber of Commerce. But in the data set these numbers are 90 restaurants and under 30 for health places.
These are two examples and I think this incompleteness is the case for most of the amenities in the data set. One possible solution for this problem is comparing these amenity data with Chamber of commerce or Yelp data sets.

Another quality problem I found in the data set is lack of *uniformity*.
Every user put a node or a way in a manner he/she likes. All users need a uniform template for each kind of data point. For example, there is one key as 'Category' and I believe it means 'amenity'. But because there is no clear definition, the user puts a new key. This kind of keys creates query problems and inaccurate results.
With cross referencing this data set with reliable data sets and creating uniform fields for each kind of data point, I think the quality of this data set will improve dramatically.

The benefits of this suggestion are:

1. Create a blue print for data points that easy to follow
2. Create a standard for new data points

On the other hand, the anticipated problems are:

1. Many people don't like instructions and maybe don't participate in the project at all
2. Every standard will eliminate innovation and creativity and we will lose some useful abnormality
3. It will create something like other maps and not an open source and active map

Another problem is limited number of contributors.
I am suggesting a plan for partnership between Pokémon and Open Street Map in this area or maybe in all areas. Santa Monica is one of the best visiting attractions in Los Angeles and there is a huge opportunity for completing and expanding the data set and its accuracy because of large number of tourists this area has.
In 2015, number of visitors was 8.3 million! If by any plan, they engaged in putting data in Open Street Map, this area will be one of the most accurate and data rich areas between all regions on the Map.

The benefits of this participation between Pokémon and Open Street map should be:

1. Many young people and millennials will commit to the game and number of users will increase drastically
2. Based on these new users work, the map will contain many details that other ordinary maps do not contain; Very detailed points like stuff located in a park

On the other hand, the anticipated problems related to this idea are:

1. Getting a deal with Pokémon should be hard and takes effort
2. It will change the type of users and the whole result is not clear
3. Marketing costs for reaching to new users is high