P3 OPEN STREET MAP DATA CASE STUDY

PROBLEMS ENCOUNTERED IN THE DOWNLOADED MAP

STREET NAME ABBREVIATION INCONSISTENCY ISSUE

After initially downloading a small sample size of the San Jose area and running it against data.py file, to check if there is any street name inconsistency issue, via the code below:

```
import re

from collections import defaultdict


str_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)


standard_strname = ["Avenue", "Boulevard", "Commons", "Court", "Drive",
"Lane", "Parkway",

                         "Place", "Road", "Square", "Street", "Trail"]


pair = {'Ave'  : 'Avenue',

           'Blvd' : 'Boulevard',

           'Dr'   : 'Drive',

           'Ln'   : 'Lane',

           'Pkwy' : 'Parkway',

           'Rd'   : 'Road',

           'Rd.'   : 'Road',

           'St'   : 'Street',

           'street' :"Street",

           'Ct'   : "Court",

           'Cir'  : "Circle",

           'Cr'   : "Court",

           'ave'  : 'Avenue',

           'Hwg'  : 'Highway',
           'Hwy'  : 'Highway',

           'Sq'   : "Square"}




def audit_street(datafile):

    datafile = open(datafile, "r")

    strtype = defaultdict(set)

    for line, elem in ET.iterparse(datafile, events=("start",)):


        if elem.tag == "node" or elem.tag == "way":

            for tag in elem.iter("tag"):

                if tag.attrib['k'] == "addr:street":

                    strsearch = str_type_re.search(tag.attrib['v'])

                    if strsearch == 1:

                        street_type = strsearch.group()

                        if street_type not in standard_strname:

                            strtype[street_type].add(strname)

    return strtype


strtype = audit_street(data)


pprint.pprint(dict(strtype))





#Update the street names

def update_name(name, pair, regex):

    regsearch = regex.search(name)

    if regsearch == 1:

        street_type = regsearch.group()
        if street_type in pair:

            name = re.sub(regex, pair[street_type], name)


    return name


for street_type, ways in strtype.iteritems():

    for name in ways:

        standarized_name = update_name(name, pair, str_type_re)

        print name, "=>", standarized_name
```

which gave out:

```
Real => Real

Sorrento => Sorrento

Marino => Marino

Plaza => Plaza

Terrace => Terrace

Way => Way

Ave => Avenue

Circle => Circle

East => East

Loop => Loop
```

It seems that this issue has been corrected. For details, please check the output json file where all the street names have been standardized.

ZIP CODE INCONSISTENCY ISSUE

Similar, I also checked the zip codes in the map data of San Jose to see if every one was correct, by using the following codes:

from collections import defaultdict

```
#define the creteria to judge if the zip code is not good
def zCode(NVZCode, zipcode):
    doubleDigi = zipcode[0:2]

    if not doubleDigi.isdigit():
        NVZCode[doubleDigi].add(zipcode)

    elif doubleDigi != 95:
        NVZCode[doubleDigi].add(zipcode)



#check the zip code one by one
def audit_zipcodes(osmfile):
    osm_file = open(osmfile, "r")
    NVZCode = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if tag.attrib['k'] == "addr:postcode":
                    zCode(NVZCode,tag.attrib['v'])

    return NVZCode
#collect the invaid zip codes and print them out
SJzCodes = audit_zipcodes(data)
pprint.pprint(dict(SJzCodes))
```

We can notice there were some zip codes with the inappropriate zip codes, such as

```
94': set(['94086', '94087']),

 '95': set(['95014-0434',

           '95014-0537',

           '95014-0547',

           '95014-0568',

           '95014-1126',

           '95014-1622',

           '95014-1627',

           '95014-1665',

           '95014-1802',

           '95014-1937',

           '95014-2008',

           '95014-2015',

           '95014-2039',

           '95014-2105',

           '95014-2137',
```

To resolve this problem, I also applied zip code updating function, by:

```
#Update zip codes

#Define a function about how to change the invaid zip codes to valuesid
zip codes

def updateZP(zipcode):

    IsNum = re.findall('[a-zA-Z]*', zipcode)

    if IsNum:

        IsNum = IsNum[0]

    IsNum.strip()

    if IsNum == "CA":

        ZCprocess = (re.findall(r'\d+', zipcode))

        if ZCprocess:
            if ZCprocess.__len__() == 2:

                return (re.findall(r'\d+', zipcode))[0] + "-"
+(re.findall(r'\d+', zipcode))[1]

            else:

                return (re.findall(r'\d+', zipcode))[0]

#Update the invaid codes one by one

for street_type, ways in SJzCodes.iteritems():

    for name in ways:

        appro_ZP = updateZP(name)

        print name, "=>", appro_ZP
```

which gave out the corrections as:

```
95014-4306 => None

95014-2008 => None

95014-232 => None

95014-3869 => None

95014-4155 => None

95070 => None

95014-2962 => None

95014-4651 => None

95014-3603 => None

95014-3602 => None

95014-5148 => None

95014-3666 => None

95037-4326 => None
```

DATA OVERVIEW AND ADDITIONAL IDEAS

After the data has been transformed to json file, I did some more exploration on the data:

CHECK FILE SIZES

```
#check file sizes

import os
```

```
print 'The original OSM file is {}
MB'.format(os.path.getsize(data)/1.0e6) # convert the unit bytes to
megabytes

print 'The JSON file is {} MB'.format(os.path.getsize(data +
".json")/1.0e6) # convert the unit bytes to megabytes
```

```
'bounds': 1,

 'member': 14715,

 'nd': 1569017,

 'node': 1343200,

 'osm': 1,

 'relation': 1461,

 'tag': 703568,

 'way': 178165}

1344

The original OSM file is 297.000869 MB

The JSON file is 340.715247 MB
```

NUMBER OF DOCUMENTS

```
#Check number of documents

SJ.find().count()
```

Which turned to be:

```
1121857
```

NUMBER OF WAYS

```
#Check number of Nodes and Ways

print "The total counts of nodes:",SJ.find({'type':'node'}).count()

print "The total  counts of ways:",SJ.find({'type':'way'}).count()
```

which told us:

```
The total counts of nodes: 99824

The total  counts of ways: 12723
```

NUMBER OF UNIQUE USERS

```
#check number of unique users

len(SJ.distinct('created.user'))
```

```
which gave out:

1028
```

FURTHER DATA EXPLORATION

After understanding several fundamental parameters of the data, such as number nodes, ways, file sizes, and the number of contributing users, I am interested in knowing some exciting data about this city, San Jose. I did two researches on it.

FIND OUT THE TOP 5 CUISINE IN SAN JOSE

```
#Find out the top 5 cuisine in San Jose

cuisines = SJ.aggregate([{"$match":{"amenity":{"$exists":1},

                                 "amenity":"restaurant",}},

                      {"$group":{"_id":{"Food":"$cuisine"},

                                 "count":{"$sum":1}}},

                      {"$project":{"_id":0,

                                  "Food":"$_id.Food",

                                  "Count":"$count"}},

                      {"$sort":{"Count":-1}},

                      {"$limit":6}])

print(list(cuisines))
```

from the calculated results, we understand the people in San Jose prefer:

Ranking	Style
1	Mexican
2	Chinese
3	Vietnamese

4	Pizza
5	Japanese

This might be related to the population ratio of San Jose, as people usually like their own style of food.

FIND OUT THE TOP 5 POST CODES IN SAN JOSE

```
#Find out the top 5 post codes in San Jose

Pcode = SJ.aggregate( [

   { "$match" : { "address.postcode" : { "$exists" : 1} } },

   { "$group" : { "_id" : "$address.postcode", "count" : { "$sum" : 1} } },

   { "$sort" : { "count" : -1}},

     {"$limit":5}] )

print(list(Postcode))
```

In summary of this output, we have:

Ranking	Style
1	95014
2	95070
3	94087
4	94086
5	95051

CONCLUSION

MY IDEA TO IMPROVE THE MAP DATA QUALITY

After this review of the data it’s obvious that the data of San Jose area is almost clear, as I believe it has been well cleaned over and over again by lots of contributing users. To make mapzen’s data even better, I hope the mapzen may set up a symmetrical standards for their map data format, for all the participants to follow or at least refer to as a preferred guide line. Also, mapzen may also introduce a function to automatically examine the data entered or updated by the participants, to make sure the data formats are following the rules.

CONSIDERATION ON MY IDEA

The benefits of my proposed idea is that the data in Mapzen could be even more accurate, especially for users who rely on it for GPS navigation and searching for an important location, to save their time and trip expenses.

Never the less, such idea cannot be realized without great efforts from huge number of participants. As mapzen is an free open resource, this could be really be a challenge for it.