Project 3
Data Wrangling with MongoDB

Derek Gurney

Map Area: Vancouver, BC, Canada
https://mapzen.com/data/metro-extracts/#vancouver-canada

1. Problems encountered in my map

An inspection of the data revealed several problems with the map data: abbreviated, inconsistent, or misspelled street types; misspelled city names; inconsistent province name; and misformatted postal codes. All of these were cleaned programmatically; additional issues are discussed below.

Abbreviated, inconsistent, or misspelled street types

Street types in the data set were abbreviated; for example, “Street” was represented as “St.” The abbreviations were also inconsistent, with “Street” sometimes abbreviated as “St” and sometimes as “St.” Lastly, street types were sometimes misspelled; for example “venue” instead of “Avenue.” These errors are captured in the following mapping, which is used in the code to update street names (shown at the end of this section).

```
mapping_street = { "Blvd": "Boulevard",
            "St": "Street",
            "St.": "Street",
```

```
            "Ave": "Avenue",
            "Rd.": "Road",
            "Steet": "Street",
            "street": "Street",
            "venue": "Avenue",
            "Broughton": "Broughton Street",
            "Jervis": "Jervis Street",
            "Jarvis": "Jervis Street"
```

Misspelled city names

“Vancouver” was sometimes spelled as “Vancovuer” and was sometimes uncapitalized. These errors are captured in the following mapping, which is used in the code to update city names (shown at the end of this section).

```
mapping_city = { "Vancovuer": "Vancouver",
            "vancouver": "Vancouver",
```

Inconsistent province name

“British Columbia” was alternately represented as “BC”, “B.C.”, and “bc”.

```
mapping_prov = { "BC": "British Columbia",
            "B.C.": "British Columbia",
            "bc": "British Columbia",
```

Misformatted postal codes

Postal codes in Canada are formatted as “LetterNumberLetter NumberLetterNumber”, but many of the postal codes were represented either without capitalization or without the middle space. The code uses regular expressions to find these situations and create a mapping between the misformatted postal codes and the correct version. As with the other errors, this mapping is used to update postal codes.

```
postalcode_valid_except_caps = re.compile(r'[a-z][5|6|7][a-z][ ][0-9][a-z][0-9]$')
postalcode_valid_except_spacing_and_possibly_caps =
re.compile(r'[A-Za-z][5|6|7][A-Za-z][0-9][A-Za-z][0-9]$')

def audit_postal_code(postal_codes, postal_code):
    '''If the postal code is valid but for specific formatting issues, fix formatting.'''
    if postalcode_valid_except_caps.search(postal_code):
        postal_codes[postal_code].add(postal_code)
```

```
        mapping_postalcodes[postal_code] = postal_code.upper()
    elif postalcode_valid_except_spacing_and_possibly_caps.search(postal_code):
        postal_codes[postal_code].add(postal_code)
        mapping_postalcodes[postal_code] = postal_code[0:3].upper() + " " +
postal_code[3:6].upper()
    else:
        pass
```

The following code uses the mappings to correct incorrect entries while converting the file from .osm format into a form that can be processed by MongoDB

```
def update_name_with_mapping(name, mapping):

    for key, value in mapping.iteritems():
        key_at_end = re.escape(key) + r"$"
        name = re.sub(key_at_end, value, name)
    return name

def shape_element(element):
# the code below is the relevant snippet from this function

if 'k' in child.attrib: #make sure that there is a key tag
    if  re.search(problemchars, child.attrib['k']):
        print "k Has problem chars: ", child.attrib['k']
    elif re.match('addr:',child.attrib['k']): #if k tag starts with addr:
        if child.attrib['k'].count(':') > 1: #if k tag has 2 colons, ignore
            pass
        else:
            temp = child.attrib['k'].split(':')[1]
            if temp == 'street':
               this_address[temp] = update_name_with_mapping(child.attrib['v'],mapping_street)
            elif temp == 'city':
               this_address[temp] = update_name_with_mapping(child.attrib['v'], mapping_city)
            elif temp == 'province':
                this_address[temp] = update_name_with_mapping(child.attrib['v'], mapping_prov)
            elif temp == 'postcode':
                this_address[temp] = update_name_with_mapping(child.attrib['v'],
                 mapping_postalcodes)
                else:
                   this_address[temp] = child.attrib['v']
```

Unaddressed issues

A number of other issues were left unaddressed: city names that are outside of the map area (e.g. Dresden), provinces outside map area (e.g. Ontario), and incomplete postal codes (e.g. V6C).

2. Overview of the Data

This section contains basic statistics about the dataset, the MongoDB queries used to gather them

File sizes

Vancouver_canada.osm: 175.8 MB
Vancouver_canada.osm.json: 203.3 MB

Number of documents

```
db.yvr.find().count()
```

3235218

Number of nodes

```
db.yvr.find({"type":"node"}).count()
```

2772348

Number of ways

```
db.yvr.find({"type":"way"}).count()
```

462801

Number of unique users

```
db.yvr.distinct("created.user").length
```

1147

Number of cities

```
db.yvr.distinct("address.city").length
```

24

Number of documents with amenity = “cafe”

```
db.yvr.find({"amenity":"cafe"}).count()
```

353

Number of Crematoriums

```
db.yvr.distinct( "id" ,{"amenity" :"Crematorium"}).length
```
1

Number of Starbucks

```
db.yvr.find({"name":/^Starbucks/}).count()
```
81

Number of Tim Hortons, a competing chain

```
db.yvr.find({"name":/^Tim Hortons/}).count()
```
23

How are Tim Hortons classified?

```
db.yvr.aggregate(
    [ { $match : { name : /^Tim Hortons/,{ $group : { _id : "$amenity" , number : {$sum: 1}} ]
);
```
Cafe : 14
Fast_food: 9

Top 10 amenities

```
db.yvr.aggregate(
    [
    { $match : {amenity: {$exists: true}},
    { $group : { _id : "$amenity" , number : { $sum : 1}} ,{$sort:{number:-1}},{$limit:10}
    ]
);
```

parking: 1020
bench: 659
restaurant: 634

cafe: 353
fast_food: 287
bicycle_parking: 227
post_box: 202
bank: 148
school: 138
toilets: 104

3. Other ideas about the dataset

The map data can provide additional insights about the geography and economics of the area. Continuing the comparison of Starbucks and Tim Hortons, it would be interesting to compare how the two chains locate themselves:

- Does Starbucks locate in areas with younger people?
- Are Tim Hortons more likely to be located near highways?
- How close are Starbucks located together?

Further to this question, it would be interesting to know the other top competitors in the market.

Top 10 cafes in Vancouver, including Tim Hortons, based on number of stores

First attempt, revised below

```
db.yvr.aggregate(
    [ { $match : {$or:[{amenity:"cafe"},{name:/^Tim Hortons/}]},{ $group : { _id : "$name" , number : { $sum : 1}} ,{$sort:{number:-1}},{$limit:10}]
);
```

Starbucks: 61
Tim Hortons: 23
Starbucks Coffee: 19
Blenz Coffee: 14
Null: 12
JJ Bean: 8
Blenz: 6
Caffè Artigiano: 3
Cafe Artigiano: 3
Waves Coffee House: 3

Since there are non-standardized names and some null entries, I updated the database and modified the query:

```
db.yvr.updateMany(
   { name: "Starbucks"},
```

```
   {
     $set: { name: "Starbucks Coffee"},
  )

db.yvr.updateMany(
   { name: "Blenz"},
   {
     $set: { name: "Blenz Coffee"},
 )

db.yvr.updateMany(
   { name: "Cafe Artigiano"},
   {
     $set: { name: "Caffè Artigiano"},
 )

db.yvr.updateMany(
   {$and:[{name: /^Waves/},{amenity:"cafe"}]},
   {
     $set: { name: "Waves Coffee House"},
 )

db.yvr.updateMany(
   { name: "JJBean"},
   {
     $set: { name: "JJ Bean"},
 )

db.yvr.aggregate(
    [
    { $match : { $and: [{$or:[{amenity:"cafe"},{name:/^Tim Hortons/}]},{name: {$exists:
true}]},
    { $group : { _id : "$name" , number : { $sum : 1}} ,{$sort:{number:-1}},{$limit:10}
    ]
);
```

Starbucks Coffee: 80
Tim Hortons: 23
Blenz Coffee: 20
JJ Bean: 10
Waves Coffee House: 9
Caffè Artigiano: 6
Bean Around The World: 3
Sciué: 2
Musette Caffè: 2
Beyond Coffee: 1

Combined with user data, the map data can also provide insights into how errors are introduced in the map and how they might be prevented. Questions in this vein would include:

- Do most new users create errors when they are new and then improve, or do only some users create most of the errors, both while they are new and later? The answer to this question would inform how to best allocate training resources to new users. If everyone makes mistakes when they are new, then everyone should receive training. However, if only some users create errors, then effort should be made to find new users making errors and target training to them. The answer to the question of who makes mistakes and when may also suggest editing policy: limit the ability of all new users to make changes from the beginning, or limit new users' ability to make changes only after they've made a certain number of errors.
- What types of geographical entities are most associated with errors? Ways or nodes? Addresses or names? Amenity classification or location? Answers to these questions would help site administrators develop "guard rails" that limited errors while encouraging participation. For example, as shown above, the "Coffee" in cafe names is often omitted (e.g. Starbucks vs Starbucks Coffee), so one implementation of this policy would be to add a prompt whenever a cafe is added to the map with the words "Do you need to add `Coffee' to the name?"

Before implementing anything that limited new users' editing privileges or made adding certain amenities more difficult, the benefits in terms of fewer errors would have to be balanced against potential costs.

One cost of limiting users' ability to change the map is that it frustrates users and discourages them from making updates. If updates decrease enough, the map may end up having fewer user-generated errors but with even more errors due to being out of date.

Another cost of frustrating users is that it discourages them from adding to the map in the first place. In the case, the map may end up being more correct, due to fewer errors, but less accurate, due to fewer objects on the map.

A cost of enacting "guard rails" would that they would likely annoy users and thus decrease participation. Further, what an administrator might consider an error may turn out to be the best way to present data, given the idiosyncrasies of the facts on the ground. After all, the map is not the territory.