Problems encountered in your map

1. There are many systematic and accidental inconsistencies in the data. For example in street types, following examples were observed in the data:
   'Terace' instead of 'Terrace'
   'Ave.' instead of 'Avenue'
   St./St/STREET/street instead of 'Street'

   Also, There were many examples that used abbreviated orientation. following examples were observed in the data:
   W./W instead of 'West'
   N./N instead of 'North'

   I used mapping for this problem:

```
mapping_Street = { "St": "Street",
            "St.": "Street",
            "STREET":"Street",
            "street":"Street",
            "Rd.": "Road",
            "Ave.": "Avenue",
            "ave": "Avenue",
            "Ave": "Avenue",
            "Dr.": "Drive",
            "Dr":"Drive",
            "Pkwy":"Parkway",
            "Blvd.": 'Boulevard',
            "Blvd": 'Boulevard',
            "blvd": 'Boulevard',
            "BLVD": 'Boulevard',
            "BLVD.": 'Boulevard',
            "Cres": "Crescent",
            "Cres.": "Crescent",
            "Grv": 'Grove',
            "Grv.": 'Grove'}
mapping_Orientation = { "N.": "North",
             "N": "North",
            "S.": "South",
            "S": "South",
```

"E.":"East",
"E":"East",
"W.": "West",
"W": "West"}

Ref: https://www.canadapost.ca/tools/pg/manual/PGaddress-e.asp?ecid=murl10006450
Ref: http://www.fnesc.ca/Attachments/BCeSIS/PDF's/addressing_guide-e.pdf

Based on the above reference there are so many other street types and abbreviation more than sixty street types and each can come in different abbreviations! I didn't use the most exhaustive list, however I used the most common types of streets as I have mentioned above in mapping_steet, mapping_orientation dictionaries and clean_street_type function!

2. To extract the street types, I had some difficulties:
   - The format of streets with orientation in Canada is like Steet_Name Street_Type Orientation; for example Yonge Street North. Therefore, it is not always the case the last word of addr:street value would be the street type.
     I used orientation_list to validate the last word of street names to check if it is orientation. If it is orientation then the second last word will be a candidate for street type.
     Orientations = ['North', 'West', 'East', 'South']
   - Furthermore, there are so many types of streets which is also different from the case that we investigated in class (Chicago). Here is some common types of streets in the data that I extracted:
     expected_street_types = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road", "Trail", "Parkway", "Commons", "Crescent", 'Terrace', 'Way', 'Circle', 'Sideroad', 'Line', 'Grove', 'Gate', 'Gardens']
   - Looking at street names, I found out several cases that no street types are mentioned for the street name. I did investigate more to see if there is a systematic solution for this but I couldn't find one. Even for some values I did search on google map, and I still couldn't find specific street type for those street names.
   - Furthermore as convention for some streets the naming is not like the case that street types go at the end or the street name. For example 'Highway 7' or 'Line 25' which are correct format of address. So for such cases one simple solution could be to extract the first word as a candidate for street types when street names end with number.
3. Postal codes in Canada are in format of "X1Y 2Z3":
   - 6 character plus one space in the middle
   - Capital letters
   - Sequence of letters and number one by one

In the dataset I noticed there are many codeposts that are not in the above format. For many of them I wrote piece of code to systematically clean them. More specifically, I found following type of examples that I pick and clean in my code using clean_postcode function:

Some of bad formatted examples in the dataset are as follows:

M2n 3e3

m2n 3e3

m2n3e3

4. There are many attribution in the openstreetmap with value FIXME that should be manually corrected or filled.
   These are mostly ‘k’ tags (for some it is ‘v’ tag). After doing more investigation, I found out that for many cases users used this attribute to signal for probable wrong location. Or the users is not sure that the location, address, amenity or etc exists/moves/changed or etc.

5. When querying for cities, there are many cities displayed other than Toronto. Some of the cities I know are definitely not in Toronto, in fact it is more than 100km far from Toronto. Also, many cities with only one node which may suggest that there should be a mistake. For some of the latter category, it is the neighborhood name that has been placed in the city attribute. Solution: It needs manual intervention to look up the position and find out the city. Or for the ones that we think there might be a mistake, it is also a good idea to compare it with other Map sources to validate the city name.

Overview of the Data

As my database, I worked on data for vicinity of Toronto, Canada. The file size before uncompressing is +70 MB.

I used the following regular expressions to check ‘k’ attributes of data:
lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')
other → the rest

There are many values that would fall under ‘other’:
Keys: {'problemchars': 12, 'lower': 696633, 'other': 8924, 'lower_colon': 1354015}
After some research I found out about many regional specification of data. For example attributes such as 'canvec' and 'geobase' that relates to map data at Canada.

In terms of tag variety, below is the stats about number of common tags in the data:

{'node': 5322966, 'tag': 2622762}

Also using MongoDB:

Number of documents

```
>>db.map.find().count()
5322967
```

Number of nodes

```
>> db.map.find({"type":"node"}).count()
1471349
```

Number of ways

```
>> db.map.find({"type":"way"}).count()
628940
```

Number of unique users

```
>> len(db.map.distinct("created.user"))
1498
```

Top 3 contributing user

```
>>db.map.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":3}])
```

1. {u'_id': u'andrewpmk', u'count': 4072588}
2. {u'_id': u'MikeyCarter', u'count': 488079}
3. {u'_id': u'Kevo', u'count': 353145}]

#Top 10 Amenities

```
>> db.map.aggregate([{"$match":{"amenity":{"$exists":1}}},
{"$group":{"_id":"$amenity","count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}])
```

1. {u'_id': u'parking', u'count': 22099}
2. {u'_id': u'fast_food', u'count': 2635}
3. {u'_id': u'school', u'count': 2561}
4. {u'_id': u'restaurant', u'count': 2352}
5. {u'_id': u'place_of_worship', u'count': 1815}
6. {u'_id': u'bench', u'count': 1813}
7. {u'_id': u'post_box', u'count': 1665}
8. {u'_id': u'cafe', u'count': 1364}
9. {u'_id': u'bank', u'count': 1086}

10. {u'_id': u'fuel', u'count': 991}

Other ideas about the datasets

1. The contribution of top user in the dataset is more than 60%. This may suggest that dataset lack validity or completeness.
2. There are many values that are tagged as FIXME, for these items which are mostly related to location or addresses or amenities it can be a good idea to validate them with other map resources.
3. There are abbreviation in street names that are not easy to correct. One example that was obvious for me was using St. instead of Saint or Sainte; there is no easy solution for that as once should decide if the street name is feminine or masculine and then he will be able to decide whether to use Sainte or Saint. I left it as is.
4. We can cross reference them with other address providers. For example, here in Canada, Canada Post has databases of more than 10 million addresses which keeps updated regularly. We can use this database to cross reference with ours to update the address names, amenity or etc in openstreetmaps
5. There are some mistakes in data points. I found out there are many nodes with different city name that are far from Toronto. One explanation is that user included wrong data points to the dataset. Another could be that user assigned wrong city name to the data point. map resources. For example, I observed a data point with city_name of Calgary which is even in another state