1. Problems Encountered in the Map

After analyzing the small map of Central, NY region I encountered three following problems with the data:

i) Multiple naming conventions for State highways and County roads (For example, State Highway 79, State Route 91, County Road 509, County Route 16)
(Run code: improveStreetNames.py)

ii) Inconsistent state abbreviations (for example, New York, NY, ny)
(Run code: auditStateAbbre.py)

iii) Multiple formats of phone numbers (for example, 607 241 3804, 607.882.2333, 6072725275, +16073301000)
(Run code: improvePhoneKeyValue.py)

Multiple road conventions

Apart from improving name conventions for street names (e.g., st=>Street, Ave=>Avenue, Ext=>Extension), as done in problem exercise. Additionally, I also found inconsistencies in naming for Street Highways and County Roads. Following are the examples of corrections made to make them consistent.

County Road 153 => County Road 153
County Route 64 => County Road 64
State Highway 38A => State Highway 38A
State Route 26 => State Highway 26

Inconsistent State abbreviations

I observed in some of the “addr” tags the state field was inconsistent. In most of the cases, the state was abbreviated as NY for New York and PA for Pennsylvania, but there were instances where either the state was not abbreviated or the abbreviations were in small letters. Following are the corrections applied in such cases.

New York => NY
ny => NY

Phone number format correction

I found a lot of inconsistencies in phone numbers. I used a regular expression to identify correct formats of US phone numbers.

```
phone_re = re.compile(r'((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}')
```

All the other formats which did not match the above regular expression was corrected to have a consistent format. Following are some of the examples of the corrections applied.

315 497 2222 => 315-497-2222
607.882.2333 => 607-882-2333
6072736464 => 607-273-6464

+16073301000 => 607-330-1000

There were some other instances where the phone numbers were invalid. For example 570-882-162 (9 digits instead of 10), and http://www.coldwatercreek.com/ (an url instead of phone number). In such cases the field was left blank.

2. Overview of Data

This section will explore some basic exploratory statistics of data. The data was analyzed using MongoDB queries.

File Sizes

fingerLakeBinghamtonRegion.osm --- 151 MB
fingerLakeBinghamtonRegion.json --- 165 MB

Number of Documents

```
> db.fingerLakeBinghamtonRegion.find().count()
744798
```

Number of nodes

```
> db.fingerLakeBinghamtonRegion.find({"type":"node"}).count()
692087
```

Number of ways

```
> db.fingerLakeBinghamtonRegion.find({"type":"way"}).count()
52659
```

Number of Unique amenities

```
> db.fingerLakeBinghamtonRegion.distinct("amenity").length
69
```

Amenity with most numbers of nodes

```
> db.fingerLakeBinghamtonRegion.aggregate([
... {"$match":{"type":"node", "amenity":{"$exists":1}}},
... {"$group":{"_id":"$amenity","count":{"$sum":1}}},
... {"$sort":{"count":-1}},
... {"$limit":1}
... ])
{ "_id" : "school", "count" : 748 }
```

Number of schools in each county

```
> db.fingerLakeBinghamtonRegion.aggregate([
... {"$match":{"type":"node", "amenity":"school"}},
... {"$group":{"_id":"$gnis_county_id", "count":{"$sum":1}}},
... {"$sort":{"count":-1}}
... ])
{ "_id" : "101", "count" : 182 }
{ "_id" : "007", "count" : 103 }
{ "_id" : "023", "count" : 76 }
{ "_id" : "123", "count" : 65 }....
```

Here _id field refers to the gnis_county_id.

Unique water ways

```
> db.fingerLakeBinghamtonRegion.distinct("water")
[
      "lake",
      "river",
      "canal",
      "pond",
      "intermittent",
      "Bog",
      "Swimming_pool",
      "reservoir"
]
```

Size of lake water ways

```
> db.fingerLakeBinghamtonRegion.aggregate([
... {"$match":{"type":"way", "water":"lake"}},
... {"$unwind":"$node_refs"},
... {"$group":{"_id":"$name","lake_size":{"$sum":1}}},
... {"$sort":{"lake_length":-1}}
... ])
{ "_id" : "Hemlock Lake", "lake_size" : 182 }
{ "_id" : "Honeoye Lake", "lake_size" : 153 }
{ "_id" : "Canadice Lake", "lake_size" : 147 }
{ "_id" : "Owasco Lake", "lake_size" : 239 }
{ "_id" : "Conesus Lake", "lake_size" : 240 }....
```

Here I am assuming, that a bigger lake would have more number of nodes in its node_refs field. Hence, counting the number of nodes in in node_refs filed is an indirect way to estimate the size of a lake.

3. Other Ideas about the dataset

In this section, I will discuss some ideas which could potentially improve the dataset.

Missing addresses for amenities

In the dataset, there exists many amenity nodes which do not have address associated with. See following example

```
>
db.fingerLakeBinghamtonRegion.findOne({"amenity":{"$exists":1},"address.state":{"$exists":0}})
{
    "_id" : ObjectId("582f6ad017f66ca435909b10"),
    "gnis_ST_num" : "36",
    "amenity" : "bar",
    "gnis_Class" : "Populated Place",
    "name" : "Seebers Tavern",
    "created" : {
        "changeset" : "28956442",
        "user" : "jbithaca",
        "version" : "3",
        "uid" : "2422820",
        "timestamp" : "2015-02-19T13:35:55Z"
    },
    "import_uuid" : "bb7269ee-502a-5391-8056-e3ce0e66489c",
    "gnis_id" : "965536",
    "gnis_ST_alpha" : "NY",
    "pos" : [
        42.3958183,
        -75.8084643
    ],
    "ele" : "312",
    "gnis_County" : "Chenango",
    "type" : "node",
    "id" : "158604258",
    "gnis_County_num" : "017",
    "is_in" : "Chenango,New York,N.Y.,NY,USA"
}
```

There exist only 166 amenity nodes with addresses whereas 2720 of the amenity nodes do not have address fields.

```
>
db.fingerLakeBinghamtonRegion.find({"type":"node","amenity":{"$exists":1},"address.state":{"$
exists":0}}).count()
2720
>
db.fingerLakeBinghamtonRegion.find({"type":"node","amenity":{"$exists":1},"address.state":{"$
exists":1}}).count()
166
```

It would have been more convenient for further detailed analysis if the address fields were present in these nodes. I would suggest that address fields should be made required fields for nodes with amenities.

Additionally, for all amenity nodes phone numbers and business hours should also be made required fields.

Nearest way information in node

I think that it would have been convenient if each node was associated with a way id of the nearest way. I think this information can be extracted from latitude and longitude information but a field indicating the nearest accessible way in node document would have been convenient for easy analysis. Such as finding number of ways accessible from the node.

User review ratings

I think each amenity node should also be associated with a rating document which lists reviews and ratings from users so that other users using this map can know the popularity of amenities and quality of services provided.

Benefits:

The above-mentioned suggestions could have following possible benefits for users:

1. By making address and phone fields as required fields in amenity node will make sure that address and phone numbers are easily accessible to the users of maps, so that the user can easily establish communication and carry out the business with the amenity provider.
2. By providing a list of nearest ways in each node will simplify the query to extract ways reaching to the node.
3. Providing information of review ratings for each amenity will provide a means to compare the quality of service of different amenities.

Anticipated problems:

For the above-suggested improvements, it is possible that there could be some limitations which could cause problems in implementing these improvements. Following are some of the anticipated problems:

1. A user creating a node for an amenity may not have all the required information such as address, phone numbers, etc. It is also possible that these suggested improvements already exist in some other documents, which I am not aware of.
2. If we assume that the user has information about the addresses and phone, then there needs a system which would verify this information.

4. Conclusion

This report lists some of the problems encountered in the map data and how those problems were improved. The map data was cleaned for the identified problems and was converted to a json document. The json document created was then imported into MongoDB database and data was analyzed using MongoDB queries. I also presented some potential improvements that can be implemented to make data more convenient to analyze and make data more informative.