1. Problems Encountered in the Map:

After browsing through the xml version of the downloaded map, I found the following issues:

- Abbreviations and inconsistency in the street names
- Incorrect/Absence of actual city name in the address
- Inconsistent postal codes
- Inconsistent phone numbers
- Inconsistent values for “source”

Abbreviations and Inconsistencies in the street name

A look at a few samples of elements in the xml data revealed that abbreviations like “rd” was being used for Road. Also, some of the words were in lower case. I updated all such strings to appropriate words.

Example:
Before Changing: <tag k="addr:street" v="Budigere cross road" />
After Updating: <tag k="addr:street" v="Budigere cross Road" />

Incorrect/Absence of actual city name in the address

I found that the city name was incorrectly tagged in some of the elements. For instance in one case the city name was “Marathalli” which is actually the name of an area in Bengaluru. Since, the map data downloaded is for Bengaluru I updated the city name to “Bengaluru” wherever it was missing.
Example:
Before Changing: <tag k="addr:city" v="Marathhalli" />
After Changing: <tag k="addr:city" v="Marathhalli, Bengaluru" />

Also, since the name of the city has been changed from “Bangalore” to “Bengaluru” recently I observed that both versions of the name have been used through out the dataset. Hence, I changed the city name from Bangalore to Bengaluru everywhere.
Example:
Before Changing: <tag k="addr:city" v="Bangalore" />
After Changing: <tag k="addr:city" v="Bengaluru" />

Inconsistent postal codes

Few of the postal codes had spaces in between. I stripped the spaces from such postal codes.
Example:
Before Changing: <tag k="addr:postcode" v="560 001" />
After Changing: <tag k="addr:postcode" v="560001" />

Inconsistent phone numbers

Few of the phone numbers had spaces or hyphens in between. I stripped the spaces and hyphens from such phone numbers. Also, in case of multiple phone numbers a few of them were seperated by comma where as the others were seperated by a semi colon. I replaced all the seperators with colon.

Examples:
Before Changing: <tag k="phone" v="+91 80 4178 3000" />
After Changing: <tag k="phone" v="+91804178 3000" />

Before Changing: <tag k="phone" v="+91-80-22070713;+91-80-22070709" />
After Changing: <tag k="phone" v="+918022070713;+918022070709" />

Before Changing: <tag k="phone" v="+91 80 4201 2308, +91 80 4201 2309" />
After Changing: <tag k="phone" v="+918042012308,;+918042012309" />

Inconsistent values for “source”

The values for source have inconsistent naming convention. For example, in some cases the name starts with lower case whereas in other cases the name starts with an upper case letter. I changed the starting letter of all sources from lower case to upper case wherever source was “bing” and “survey”.

Example:
Before Change: <tag k="source" v="bing" />
After Change: <tag k="source" v="Bing" />

Before Change: <tag k="source" v="survey" />
After Change: <tag k="source" v="Survey" />

2. Data Overview

Item	Value	Code
City	Bengaluru	
Filename	bengaluru_india.osm	
File Size	678 MB	
Actual File Size Used	171 MB (25% of the original data)	
Json data size	198 MB	
Number of documents	952984	*mongoimport.exe --db test --collection bengaluru --file bengaluru_india.osm.json*
Number of nodes	772587	*pipeline = [{"$group":{"_id":"$type","count": {"$sum":1}}}] result = db.bengaluru.aggregate(pipeline)*
Number of ways	180397	
Number of unique users:	857	*num_users = len(db.bengaluru.distinct("created.user"))*
Top 10 users:	Top 10 users: [{u'count': 37987, u'_id': u'premkumar'}, {u'count': 32112, u'_id': u'saikumar'}, {u'count': 31319, u'_id': u'akhilsai'}, {u'count': 28698, u'_id': u'vamshikrishna'}, {u'count': 26216, u'_id': u'jasvinderkaur'}, {u'count': 25796, u'_id': u'shekarn'}, {u'count': 25277, u'_id': u'hareesh11'}, {u'count': 25150, u'_id': u'thrinath'}, {u'count': 24731, u'_id': u'praveeng'}, {u'count': 24220, u'_id': u'masthanvali'}]	pipeline = [{"$group":{"_id":"$created.user","count": {"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}] result = db.bengaluru.ag
Number of users who have update only once:	18906	*pipeline = [* *{"$group":{"_id":"$created.user","count": {"$sum":1}}}, {"$group":{"_id":"$count","num": {"$sum":1}}}, {"$sort":{"num":1}}, {"$limit":1}] result = db.bengaluru.aggregate(pipeline)*
Number of shops	577	*pipeline = [* *{"$match":{"shop":{"$exists":1}}}, {"$group":{"_id":None,"count":{"$sum":1}}},] result = db.bengaluru.aggregate(pipeline)*
Number of distinct sources	42	*pipeline = [*

		{"$match":{"source":{"$exists":1}}}, *{"$group":{"_id":"$source","count":* *{"$sum":1}}},* *{"$group":{"_id":None,"count":{"$sum":1}}},* *]* *result = db.bengaluru.aggregate(pipeline)*
Number of distinct highways	12	*pipeline = [* *{"$match":{"highway":{"$exists":1}}},* *{"$group":{"_id":"$source","count":* *{"$sum":1}}},* *{"$group":{"_id":None,"count":{"$sum":1}}},* *]* *result = db.bengaluru.aggregate(pipeline)* *print "Types of sources:"* *print list(result)*

3. Additional Ideas

- It can be seen that around GPS sources account for 9% of the total sources. This source can be exploited for improving the accuracy of the map data. We can receive gps streams from users and match these streams with ways existing in the map data. In case no match found we can conclude that the way represented by the gps data is missing in our maps. Hence, we can update our map with new ways using this procedure.
- Amenity based tagging: It can be observed from the data that out of the total set of documents around 951250 have not been assigned any amenity tag.
  Out of the rest of the nodes/ways which have assigned tags the number of essential services like fire station (4 tags) and police station (24 tags) have very few tags. Since these are essential services, a more essential amenities based approach has to be put in place.
- Many of the amenity tags are different though they form a similar group. For example, there are different tags like restaurants, cafe, confectionaries, etc which are actually subsets of a general tag “food”. It would be very helpful if we add a generic amenity tag to the nodes which form a similar group.
- Postal codes: The postal codes are not available for all the elements. We can extract postal codes for a particular element based on its area by searching through the other elements which are from the same area.

Additional data exploration

Item	Value	Code
Top 5 postal codes	[{u'count': 23, u'_id': u'560066'}, {u'count': 20, u'_id': u'560040'}, {u'count': 12, u'_id': u'560095'}, {u'count': 12, u'_id': u'560048'}, {u'count': 11, u'_id': u'560079'}]	pipeline = [{"$match":{"address.postcode": {"$exists":1}}}, {"$group": {"_id":"$address.postcode","count": {"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit": 5}] result = db.bengaluru.aggregate(pipeline)
Top 5 amenities	[{u'count': 257, u'_id': u'restaurant'}, {u'count': 184, u'_id': u'place_of_worship'}, {u'count': 144, u'_id': u'school'}, {u'count': 133, u'_id': u'bank'}, {u'count': 127, u'_id': u'atm'}]	pipeline = [{"$match":{"amenity": {"$exists":1}}}, {"$group":{"_id":"$amenity","count": {"$sum":1}}}, {"$sort":{"count":-1}} , {"$limit": 5}] result = db.bengaluru.aggregate(pipeline)

Top 5 shops	[{u'count': 91, u'_id': u'clothes'}, {u'count': 64, u'_id': u'supermarket'}, {u'count': 44, u'_id': u'bakery'}, {u'count': 33, u'_id': u'convenience'}, {u'count': 30, u'_id': u'car'}]	pipeline = [{"$match":{"shop":{"$exists":1}}}, {"$group":{"_id":"$shop","count": {"$sum":1}}}, {"$sort":{"count":-1}} , {"$limit": 5}] result = db.bengaluru.aggregate(pipeline)
Top 5 highways	[{u'count': 7209, u'_id': u'residential'}, {u'count': 2859, u'_id': u'unclassified'}, {u'count': 736, u'_id': u'bus_stop'}, {u'count': 701, u'_id': u'service'}, {u'count': 644, u'_id': u'tertiary'}]	pipeline = [{"$match":{"highway": {"$exists":1}}}, {"$group": {"_id":"$highway","count": {"$sum":1}}}, {"$sort":{"count":-1}} , {"$limit": 5}] result = db.bengaluru.aggregate(pipeline)
Top 5 sources	[{u'count': 298, u'_id': u'Bing'}, {u'count': 102, u'_id': u'Bing Sat'}, {u'count': 54, u'_id': u'GPS'}, {u'count': 46, u'_id': u'survey'}, {u'count': 18, u'_id': u'Landsat'}]	pipeline = [{"$match":{"source":{"$exists":1}}}, {"$group":{"_id":"$source","count": {"$sum":1}}}, {"$sort":{"count":-1}} , {"$limit": 5}] result = db.bengaluru.aggregate(pipeline)

Conclusion:

It can be observed that the data in the map for Bengaluru is incomplete as many of the elements have not been tagged with appropriate name/description. Though the data has been cleaned for inconsistencies and other inaccuracies there still remain other inaccuracies and inconsistencies like street name containing entire addressses or house number containing entire addresses. These can be fixed using NLP techniques.