A. Problems Encountered in the Map

I initially tried to download the Sydney metro extract but the file size is/was only 17.8 MB way below the project required minimum size of 50MB. I searched for Sydney on OpenStreetMap's search field and after some tweaking got a link to download the map file. It was only after I downloaded and saved the file that I found out the file size was about 226MB.

I have carried out my analysis based on this file. I have identified two problems that I encountered in the map and which I have tried to solve and rectify. Due to the size of the file and resulting inconsistencies I have ***not*** solved all problems instead I have focused on solving bulk of the inconsistencies in 2 areas.

Caveat: As in Lesson 6 I have restricted my analysis in respect of "way" and "node" tags only.

1. Street Name Inconsistencies

Since we have carried out improving street name exercise in lesson 6 it is natural to first focus on this potential problem. In addition to "audit.py" I wrote a function [find_st_names(filename)-road_types.py] which produces a set of all street names (restricted to "way" and "node" tags). Based on manual inspection of the set produced by this function I removed those names that appeared normal/correct ***and*** incorrectly abbreviated or spelled for example "St", "St.", "Strreet", "st" instead of street. Leaving out only those that were not making any sense. I have termed this set "WeirdStType". I have primarily relied on a document by Australia Post (refer to Barcode_hints_tips.pdf file) to check whether the nomenclature or the abbreviations used for streets and different street types are correct.

My approach towards this problem has been to try and completely rectify misspellings and incorrect abbreviations and correct some instances in "WeirdStType" set. The instances in "WeirdStType" are mostly of the type where the street name is not followed by the street type. I found this by first writing a small function [find_wrd_st_names(filename) - road_types.py] which creates a dictionary with each instance in "WeirdStType" as key and a set of "id numbers" of each map entry associated with that instance as the value for the key. By doing some manual searches in the map xml document as well as searching longitude and latitude values in Google I found out that almost all of the instances in "WeirdStType" set suffer from absent street type suffix. Since confirming the correct street type involves searching google or some other search engines ***I have not*** rectified all the instances appearing in "WeirdStType" set.

The following dictionary named "mapping" has been used to correct inconsistencies in street names and abbreviations [mapping-audit.py]

```
{"Androtis": "Bank Street",
 "Av.": "Avenue",
"Ave":"Avenue",
"Barney":"BarneyStreet",
"Berith":"BerithStreet",
"Bigge":"BiggeStreet",
"Bouldevarde":"Boulevard",
"Boulevarde":"Boulevard",
"Bvd":"Boulevard",
"Centenary":"CentenaryRoad",
"Clontarf":"ClontarfStreet",
"Corination":"Corination Street",
"Hwy": "Highway",
"Ln": "Lane",
"Pde": "Parade",
"place": "Place",
"Rd": "Road",
"Rad": "Road",
"road": "Road",
"St": "Street",
"St.": "Street",
"Strreet": "Street",
"st": "Street",
"street": "Street",
"Wolli": "Wolli Street",
"Wollit": "Wolli Street"}
```

2. Incorrect value for “type” field

After carrying out street names and abbreviation cleaning, I created a json file by running the code in data.py I uploaded the json file into Mongo DB using mongoimport.

Following the upload into the database I ran a few queries to sense check the data.

Query	Description	Result
db.sydmap.find().count()	Total documents in the db.	1187006
db.sydmap.find({"type":"node"}).count()	Total documents in db which are of “node” type.	1043366
db.sydmap.find({"type":"way"}).count()	Total documents in db which are of “way” type.	142956

The results of the last two queries do not add up to the total number of documents in the database as per the first query result.

1043366 + 142956 = 1186322

I ran the following query to check how many documents in the database are not of “node” or “way’ type.

Query	Description	Result
db.sydmap.find({"type": {"$nin":["node", "way"]}}).count()	Neither “node” nor “way” type	684

Following this query I created a json file containing all the 684 documents that have a “type” value other than “node” or “way”. I also created a dictionary from these documents with “type” value as key and number of instances of those “type” values.

```
{u'0.0': 7,
 u'2.0': 3,
 u'7.1.6': 1,
 u'9.0': 5,
 u'ALPR': 1,
 u'Travellift': 1,
 u'Tree': 1,
 u'audio': 1,
 u'audio;video': 1,
 u'beacon_cardinal': 14,
 u'beacon_isolated_danger': 1,
 u'beacon_lateral': 49,
 u'beacon_special_purpose': 1,
 u'boundary': 3,
 u'broad_leaved': 128,
 u'buoy_special_purpose': 1,
 u'bushfire': 1,
 u'butane': 1,
 u'caustic soda': 1,
 u'chain_link': 1,
 u'chimney': 4,
u'club': 1,
 u'communication': 38,
 u'conifer': 101,
 u'destroyer': 1,
 u'drain': 3,
 u'enforcement': 7,
 u'gas': 1,
 u'hangar': 4,
 u'light_major': 4,
 u'light_minor': 28,
 u'lighting': 5,
 u'lighting;communications': 1,
 u'military': 1,
 u'military hls': 1,
 u'multipolygon': 6,
 u'observation': 1,
 u'oil': 5,
 u'open_stable': 1,
 u'outdoor': 1,
 u'palm': 155,
u'palm1': 1,
u'patrol_boat': 1,
 u'pillar': 2,
 u'propane': 1,
 u'public': 3,
 u'range': 4,
 u'restriction': 1,
 u'sailboat': 1,
 u'sewage': 6,
 u'sewage vent': 1,
 u'single': 2,
 u'small_craft_facility': 4,
 u'special_needs': 1,
 u'split_rail': 2,
 u'staging_area': 1,
 u'street_lamp': 1,
 u'submarine': 1,
 u'video': 1,
 u'wall': 2,
 u'water': 55,
 u'wreck': 2}
```

Using this dictionary, the json file containing all the 684 document with a different “type” value and the original map xml document I found that these 684 documents represent those “node” or “way” data structures within the map document that have a “type” value associated with one of the “k” attributes within the element structure of “node” or “way” tag. Please refer to the following example.

```
<node id="19963044" lat="-33.8261207" lon="151.1991937" version="6" timestamp="2013-05-
23T11:55:50Z" changeset="16251272" uid="1461056" user="I_vanl">
  <tag k="highway" v="traffic_signals"/>
  <tag k="type" v="enforcement"/>
 </node>
```

Consequently when the map xml data is being parsed into the json file the value associated with the “v” attribute in respect of the corresponding “k” attribute with a “type” value overwrites the “node” value for “type” key defined in the shape_element(element) function in data.py

Accordingly, the above example is transformed into a dictionary where the “type” key has “enforcement” as value.

```
{u'created': {u'changeset': u'16251272', u'version': u'6', u'uid': u'1461056', u'timestamp': u'2013-
05-23T11:55:50Z', u'user': u'I_vanl'}, u'pos': [-33.8261207, 151.1991937], u'address': {}, u'_id':
ObjectId('54a77f2a17aa5e5b465ffb36'), u'type': u'enforcement', u'id': u'19963044', u'highway':
u'traffic_signals'}
```

After going through the json file containing the 684 non “node” or “way” types I also found that most of these instances have either a “natural” key or a “man_made” key. In order to support my findings I ran the following queries.

Number of documents which are not of “node” or “way” type and have either “natural” or “man-made” fields:

```
db.sydmap.find({"type": {"$nin":["node", "way"]}, "$or":[{"natural":{"$exists":1}},
{"man_made":{"$exists":1}}]}).count()
521
```

Number of documents which are not of “node” or “way” type and do not have “natural” or “man_made” fields:

```
db.sydmap.find({"type": {"$nin":["node", "way"]}, "natural":{"$exists":0}, "man_made":
{"$exists":0}}).count()
163
```

The above two results add to 684.
In order to ensure that “natural” and “man_made” are not overlapping I ran the following queries.

Number of tags which are not of “node” or “way” type and have “natural” attribute and do not have “man-made” attribute:

```
db.sydmap.find({"type": {"$nin":["node", "way"]}, "natural":{"$exists":1}, "man_made":
{"$exists":0}}).count()
386
```

Number of tags which are not of “node” or “way” type and have “man-made” attribute and do not have “natural” attribute:

```
db.sydmap.find({"type": {"$nin":["node", "way"]}, "natural":{"$exists":0}, "man_made":
{"$exists":1}}).count()
Out[20]: 135
```

The results of the last 2 queries add to 521.

Based on these results I came up with a solution which essentially deals with map data structures which when parsed result in dictionaries having "type" key values other than "node" or "way". Though my solution deals with all such instances however the solution is more tuned with dictionaries having "natural" or "man_made" keys ***and*** "type" keys. The solution is less than perfect for dictionaries that ***do not have*** "natural" or "man_made" keys ***and*** have "type" key values other than "node" or "way". In these instances the "type" key values other than "node" or "way" are not reflected in the final dictionary.

If there is a "natural" key in a dictionary after parsing an xml data structure then replace the value for "natural" key with a dictionary:

"natural" : {"type": value for "natural" key, "name": value for type tag in xml data structure}

If there is a "man_made" key in a dictionary after parsing an xml data structure then replace the value for "man_made" key with a dictionary:

man_made:{"type": value for "man_made" key, "purpose": value for type tag in xml data structure }

I revised the shape_element(element) function in data.py to reflect the above solution in the json document obtained after parsing the map xml document. I ran data.py to get a new json document and uploaded it into MongoDB into a new database.

Query	Description	Result
db.sydmap.find().count()	Total documents in the db.	1187006
db.sydmap.find({"type":"node"}).count()	Total documents in db which are of "node" type.	1043923
db.sydmap.find({"type":"way"}).count()	Total documents in db which are of "way" type.	143083

The results of the last 2 queries add to1187006

Results for "natural"
One of the document where the underlying xml document had a "type" attribute as well as a "natural" attribute:

```
db.sydmap.find({"natural.name":{"$ne":""},"natural.type":{"$exists":1}}).next()
```

```
{u'_id':
ObjectId('54c4d27561090ca9c2e7c1a4'),
 u'address': {},
 u'created': {u'changeset': u'7750874',
  u'timestamp': u'2011-04-03T07:36:38Z',
  u'uid': u'237525',
  u'user': u'aharvey',
  u'version': u'1'},
u'id': u'1230267658',
 u'location': u'nearmap',
 u'natural': {u'name': u'palm', u'type': u'tree'},
 u'pos': [-34.0514715, 151.1531135],
 u'source': u'survey',
 u'type': u'node'}
```

The above solution is not perfect as it treats all dictionaries with "natural" key in the same way as defined by my solution even if the underlying xml data structure does not have a "type" attribute. This will cause "natural.name" to be equal to ""

Here is a document where the underlying xml data structure did not have a type attribute:

```
db.sydmap.find({"natural.name":"","natural.type":{"$exists":1}}).next()
```

```
{u'_id':
ObjectId('54c4d22561090ca9c2e1ac54'),
 u'address': {},
 u'created': {u'changeset': u'23272461',
  u'timestamp': u'2014-06-29T15:38:54Z',
  u'uid': u'1187510',
  u'user': u'dhx1',
  u'version': u'6'},
 u'id': u'13804584',
 u'name': u'Kissing Point',
u'natural': {u'name': u'', u'type': u'cape'},
 u'place': u'locality',
 u'pos': [-33.8308506, 151.1018466],
 u'source': u'historical',
 u'source_ref':
u'http://parishmaps.lands.nsw.gov.au/mrsid/ima
ge_sid.pl?client=pmap&image=PMapMN03/140
95001.sid',
 u'type': u'node'}
```

It should be noted that this instance has a “name” key which was an attribute in the underlying xml document.

Results for “man_made”

One of the document where the underlying xml document had a “type” attribute as well as a “man_made” attribute:

```
{u'_id':
ObjectId('54c4d22861090ca9c2e1dae4'),
 u'address': {},
 u'created': {u'changeset': u'15863307',
 u'timestamp': u'2013-04-25T17:21:06Z',
 u'uid': u'128186',
 u'user': u'malcolmh',
 u'version': u'5'},
u'historic': u'building',
u'id': u'26525026',
u'man_made': {u'purpose': u'light_major',
u'type': u'lighthouse'},
u'name': u'Hornby',
u'pos': [-33.8335652, 151.2809945],
u'source': u'US NGA Pub. 111. 2010-09-02.',
u'type': u'node'}
```

Again as before the above solution is not perfect as it treats all dictionaries with “man_made” key in the same way as defined by my solution even if the underlying xml data structure does not have a “type” attribute. This will cause "man_made.purpose" to be equal to “”

Here is a document where the underlying xml data structure did not have a type attribute:

```
db.sydmap.find({"man_made.type":{"$exists":1}, "man_made.purpose":""}).next()
```

```
{u'_id':
ObjectId('54c4d22961090ca9c2e1e9dd'),
u'address': {},
u'created': {u'changeset': u'31549',
u'timestamp': u'2007-04-29T09:07:18Z',
u'uid': u'7523',
u'user': u'inas',
u'version': u'1'},
u'created_by': u'JOSM',
u'id': u'28195205',
u'man_made': {u'purpose': u'', u'type':
u'survey_point'},
u'name': u'Loftus Trig',
u'pos': [-34.0654117, 151.0440488],
u'type': u'node'}
```

I have included one more query result to show what happens when an xml data structure does not have “natural” or “man_made” keys ***and*** has “type” key values other than “node” or “way”. As mentioned earlier, in instances like this the “type” key value if other than “node” or “way” is not reflected in the final dictionary.

db.sydmap.find({"id":"1323184558"}).next()

```
{u'_id': ObjectId('54c4d27961090ca9c2e80fa2'),
 u'address': {},
 u'created': {u'changeset': u'9107813',
 u'timestamp': u'2011-08-23T21:16:32Z',
 u'uid': u'128186',
 u'user': u'malcolmh',
 u'version': u'3'},
u'id': u'1323184558',
u'location': u'nearmap',
u'pos': [-33.9848557, 151.1491798],
u'source': u'survey',
u'type': u'node'}
```

Here is the original xml document for the above json dictionary.

```
<node id="1323184558" lat="-33.9848557" lon="151.1491798" version="3" timestamp="2011-08-
23T21:16:32Z" changeset="9107813" uid="128186" user="malcolmh">
   <tag k="seamark:beacon_cardinal:category" v="east"/>
   <tag k="seamark:beacon_cardinal:colour" v="black;yellow;black"/>
   <tag k="seamark:beacon_cardinal:colour_pattern" v="horizontal"/>
   <tag k="seamark:topmark:colour" v="black"/>
   <tag k="seamark:topmark:shape" v="2 cones base together"/>
   <tag k="seamark:type" v="beacon_cardinal"/>
   <tag k="source" v="survey"/>
   <tag k="source:location" v="nearmap"/>
 </node>
```

B. Data Overview

Basic statistics about the dataset and MongoDB queries for collecting statistics about the dataset.

File	Size
syd_map.osm	226 MB
syd_map.osm.json	268 MB

1. Number of documents:
db.sydmap.find().count()
1187006

2. Number of nodes:
db.sydmap.find({"type":"node"}).count()
1043923

3. Number of ways:
db.sydmap.find({"type":"way"}).count()
143083

4. Number of unique users:
I carried out my queries through ipython console using pymongo so the equivalent query for:

```
db.sydmap.distinct({"created.user"}).length
```

is

```
len(db.sydmap.distinct("created.user"))
1369
```

There is another contrived way of getting the same answer:

```
db.sydmap.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}},{"$group":{"_id":"$cre
ated.user","count":{"$sum":1}}}])
{u'ok': 1.0, u'result': [{u'_id': None, u'count': 1369}]}
```

5. Top 1 contributing user

```
db.sydmap.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count": -
1}}, {"$limit":1}])
{u'ok': 1.0, u'result': [{u'_id': u'behemoth14', u'count': 119736}]}
```

6. Number of users appearing only once (having 1 post)

```
db.sydmap.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$group":{"_id":
"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":1}])
{u'ok': 1.0, u'result': [{u'_id': 1, u'num_users': 261}]}
```

The above query can also be tweaked to support the Top 1 contributing user query:

```
db.sydmap.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$group":{"_id":
"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":-1}}, {"$limit":1}])
{u'ok': 1.0, u'result': [{u'_id': 119736, u'num_users': 1}]}
```

C. Additional Ideas

a. Types of sources and their share in map data

While auditing the data and going through the results of the queries I got really interested in knowing how this map is put together as in what sources are used to construct "OpenStreeMap" map. The about link (http://www.openstreetmap.org/about) mentions the following:

> *"OpenStreetMap emphasizes local knowledge. Contributors use aerial imagery, GPS devices, and low-tech field maps to verify that OSM is accurate and up to date."*

Based on some queries and doing some basic counting using notepad++ I am not sure that the above statement is entirely accurate unless "aerial imagery: includes data from other search engines. ***(After checking Open Street Map Wiki it appears that Bing and Yahoo are kosher).***

1. Number of "sources" tag in the project database

```
db.sydmap.find({"source":{"$exists": 1}}).count()
80115
```

2. Number of unique “sources” type in the database

Again there are two ways of doing this

The simple way:

```
len(db.sydmap.distinct("source"))
526
```

The contrived way:

```
db.sydmap.aggregate([{"$match":{"source":{"$exists": True}}},{"$group":{"_id":"$source",
"count":{"$sum":1}}},{"$group":{"_id":"$source", "count":{"$sum":1}}}])
{u'ok': 1.0, u'result': [{u'_id': None, u'count': 526}]}
```

3. Top 10 sources

<u>Without a check for the “source” field</u>

```
db.sydmap.aggregate([{"$group":{"_id":"$source", "count":{"$sum":1}}}, {"$sort":{"count":-1}},
{"$limit":10}])

{u'ok': 1.0,
 u'result': [{u'_id': None, u'count': 1106891},
  {u'_id': u'nearmap', u'count': 21150},
  {u'_id': u'survey', u'count': 18804},
  {u'_id': u'Bing', u'count': 16133},
  {u'_id': u'bing', u'count': 4150},
  {u'_id': u'GPS survey', u'count': 2712},
  {u'_id': u'yahoo', u'count': 1684},
  {u'_id': u'yahoo_imagery', u'count': 1556},
  {u'_id': u'local_knowledge', u'count': 1374},
  {u'_id': u'ABS2011-data.gov.au', u'count': 1074}]}
```

This clearly shows that the overwhelming majority of the documents ***do not*** have a “source” field.

<u>With a check for the “source” field</u>

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

{u'ok': 1.0,
 u'result': [{u'_id': u'nearmap', u'count': 21150},
  {u'_id': u'survey', u'count': 18804},
  {u'_id': u'Bing', u'count': 16133},
  {u'_id': u'bing', u'count': 4150},
  {u'_id': u'GPS survey', u'count': 2712},
  {u'_id': u'yahoo', u'count': 1684},
  {u'_id': u'yahoo_imagery', u'count': 1556},
  {u'_id': u'local_knowledge', u'count': 1374},
  {u'_id': u'ABS2011-data.gov.au', u'count': 1074},
  {u'_id': u'PGS', u'count': 1063}]}
```

I am not sure what “nearmap’ source is as I could not find it on wiki.

Further, it will be clear from the query results there are inconsistencies about naming of the sources there are 2 “bing/Bing” sources and 2 “yahoo” sources

We can revise the query in light of this information:

```
db.sydmap.aggregate([{"$match":{"source":{"$exists":True}}},{"$group":{"_id":"$source",
"count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":12}])
```

```
{u'ok': 1.0,
 u'result': [{u'_id': u'nearmap', u'count': 21150},
  {u'_id': u'survey', u'count': 18804},
  {u'_id': u'Bing', u'count': 16133},
  {u'_id': u'bing', u'count': 4150},
  {u'_id': u'GPS survey', u'count': 2712},
  {u'_id': u'yahoo', u'count': 1684},
  {u'_id': u'yahoo_imagery', u'count': 1556},
  {u'_id': u'local_knowledge', u'count': 1374},
  {u'_id': u'ABS2011-data.gov.au', u'count': 1074},
  {u'_id': u'PGS', u'count': 1063},
  {u'_id': u'Yahoo maps', u'count': 861},
  {u'_id': u'knowledge', u'count': 696}]}
```

This result shows another “Yahoo” source. I decided to stop digging at this point.

4. Top 10 sources and their contribution statistic

```
db.sydmap.aggregate([{"$match":{"source":{"$exists":True}}},
{"$group":{"_id":{"source":"$source"}, "count":{"$sum":1}}},
{"$group":{"_id":"null", "srcs":{"$push":{"src":"$_id.source", "subt":"$count"}},
"tot":{"$sum":"$count"}}},
{"$unwind":"$srcs"},
{"$project":{"_id":0,"src":"$srcs.src","subt":"$srcs.subt","tot":"$tot","ratio":{"$divide":["$srcs.subt",
"$tot"]}}}},
{"$sort":{"ratio":-1}},
{"$limit":10}])
```

```
{u'ok': 1.0,
 u'result': [{u'ratio': 0.2639955064594645,
   u'src': u'nearmap',
   u'subt': 21150,
   u'tot': 80115},
  {u'ratio': 0.23471260063658492,
   u'src': u'survey',
   u'subt': 18804,
   u'tot': 80115},
  {u'ratio': 0.20137302627473008,
   u'src': u'Bing',
   u'subt': 16133,
   u'tot': 80115},
  {u'ratio': 0.05180053672845285,
   u'src': u'bing',
   u'subt': 4150,
   u'tot': 80115},
  {u'ratio': 0.033851338700617864,
   u'src': u'GPS survey',
   u'subt': 2712,
   u'tot': 80115},
  {u'ratio': 0.021019784060413157,
   u'src': u'yahoo',
   u'subt': 1684,
   u'tot': 80115},
  {u'ratio': 0.01942208075890907,
   u'src': u'yahoo_imagery',
   u'subt': 1556,
   u'tot': 80115},
  {u'ratio': 0.01715034637708294,
   u'src': u'local_knowledge',
   u'subt': 1374,
   u'tot': 80115},
  {u'ratio': 0.013405729264182738,
   u'src': u'ABS2011-data.gov.au',
   u'subt': 1074,
   u'tot': 80115},
  {u'ratio': 0.01326842663670973,
   u'src': u'PGS',
   u'subt': 1063,
   u'tot': 80115}]}
```

b. Additional exploration using MongoDB queries

1. Number of documents with “amenity” field

```
db.sydmap.find({"amenity":{"$exists":True}}).count()
11579
```

2. Top 10 amenities in the database

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

{u'ok': 1.0,
 u'result': [{u'_id': u'parking', u'count': 2746},
  {u'_id': u'school', u'count': 853},
  {u'_id': u'bench', u'count': 751},
  {u'_id': u'cafe', u'count': 566},
  {u'_id': u'restaurant', u'count': 540},
  {u'_id': u'toilets', u'count': 520},
  {u'_id': u'fast_food', u'count': 463},
  {u'_id': u'pub', u'count': 424},
  {u'_id': u'place_of_worship', u'count': 407},
  {u'_id': u'drinking_water', u'count': 402}]}
```

3. Biggest religion

```
db.sydmap.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
{"$group":{"_id":"$religion", "count":{"$sum":1}}}])
```

```
{u'ok': 1.0,
 u'result': [{u'_id': u'christian', u'count': 323},
  {u'_id': None, u'count': 59},
  {u'_id': u'muslim', u'count': 7},
  {u'_id': u'sikh', u'count': 6},
  {u'_id': u'buddhist', u'count': 5},
  {u'_id': u'jewish', u'count': 2},
  {u'_id': u'cao_dai', u'count': 1},
  {u'_id': u'bahai', u'count': 1},
  {u'_id': u'hindu', u'count': 1},
  {u'_id': u'masonic', u'count': 1},
  {u'_id': u'scientologist', u'count': 1}]}
```

It appears that there are 59 places of worship without a designated religion – something I really wish could happen in real life all the time – which I think are omissions needing to be fixed.

Conclusion

After going through the data and doing random checks on google maps I think that the available data itself is pretty accurate. However I think the contributors need to consider following two points when putting together the data:

- Rules for defining a particular place or objects should be observed strictly by all contributors especially when it comes to "natural" and "man_made" objects.
- There should be clarity about sources used for putting together the data.

Lastly I could not figure out the meaning of "nearmap" source.