OpenStreetMap Sample Project Data Wrangling with MongoDB

Guo Yading

1. Problems Encountered in the Map

1.1 Running "audit.py"

I downloaded the map of Beijing in China in xml format. After a little digging, I run my audit.py against the xml file and print the result in an organized way.

In [1]:

```
execfile( "audit.py")
```

```
************************************************************
Problem 1: inconsistency in keys
Unicode keys in the "tag" element are:
[u'\u758f\u6563\u4eba\u6570', u'\u9ec4\u5357\u82d1\u5c0f\u533
a', u'\u8f66\u5e93']
------------------------------------------------------------
Strings containing hyphen are:
['name:bat-smg', 'name:roa-rup', 'x-shop', 'x-highway', 'x-na
me', 'name:cbk-zam', 'name:zh-simplified', 'name:roa-tara',
'name:be-x-old', 'name:zh-yue', 'name:zh-traditional', 'nam
e:zh-min-nan', 'name:zh-classical']
------------------------------------------------------------
Strings containing uppercase letters or abbreviated words ar
e:
['No.', 'currency:CNY', 'gns:UFI', 'gns:UNI', 'gns:DSG', 'gn
s:ADM1', 'FIXME']
************************************************************
```

```
************************************************************
Problem 2: colon(s) in keys
Strings containing single colon:
['ref:en', 'name:uz', 'name:ur', 'name:ug', 'payment:debit_ca
rds', 'name:ilo', 'name:ia', 'name:lez', 'name:id', 'name:i
o', 'name:is', 'name:si', 'name:nrm', 'name:br', 'name:bn',
'name:bo', 'name:bjn', 'name:be', 'name:bg', 'name:ba', 'nam
e:bcl', 'name:crh', 'alt_name:zh_pinyin', 'name:ps', 'diet:ve
getarian', 'name:cdo', 'name:sco', 'name:scn', 'alt_name:zh_p
inyin2', 'alt_name:is', 'name:pnb', 'name:ms', 'name:mr', 'na
me:my', 'name:mg', 'name:ml', 'name:mn', 'name:mi', 'name:m
k', 'name:hif', 'name:fo', 'name:fa', 'addr:place', 'fuel:cn
g', 'name:tr', 'name:tt', 'name:tk', 'name:th', 'name:pdc',
'name:tl', 'name:ta', 'name:tg', 'name:te', 'name:pms', 'nam
e:ay', 'name:az', 'ref:vi', 'name:gan', 'name:ar', 'name:am',
'name:an', 'name:af', 'alt_name:en2', 'name:za', 'name:so',
'name:sl', 'name:war', 'name:sc', 'name:sa', 'name:arz', 'nam
e:sw', 'name:sv', 'name:su', 'name:sr', 'name:sq', 'name:xm
f', 'email:reservation', 'name:tpi', 'name:hi', 'name:he', 'n
ame:bpy', 'name:hy', 'name:ht', 'name:hr', 'name:srn', 'nam
e:szl', 'name:lmo', 'addr:district', 'name:haw', 'name:hak',
'disused:amenity', 'name:xal', 'alt_name:fr', 'name:ang', 'i
s_in:iso_3166_2', 'name:ka', 'name:ko', 'name:kn', 'name:km',
'name:kl', 'name:kk', 'name:kw', 'name:kv', 'name:zh_py', 'na
me:frp', 'name:lv', 'name:lt', 'name:la', 'name:lo', 'name:e
l', 'name:eo', 'name:lij', 'name:eu', 'name:et', 'name:ro',
'name:ckb', 'name:ast', 'old_name:ku', 'fuel:diesel', 'name:s
ah', 'name:oc', 'name:os', 'toilets:disposal', 'name:ceb', 'n
ame:yi', 'name:yo', 'gns:dsg', 'name:ext', 'fuel:octane_92',
'flag:type', 'name:wuu', 'name:mwl', 'name:cy', 'name:cv', 'n
ame:cs', 'name:ca', 'name:simple', 'name:kab', 'name:da', 'na
me:dv', 'name:nah', 'name:pa', 'name:jv', 'alt_name:sv', 'nam
e:jbo', 'name:mzn', 'name:mhr', 'name:nov', 'name:pap', 'nam
e:rue', 'name:pam', 'name:qu', 'addr:unit', 'name:gl', 'old_n
ame:vi', 'fire_hydrant:type', 'name:nl', 'name:nn', 'name:n
```

```
o', 'name:na', 'name:ne', 'name:ga', 'name:gd', 'name:gn', 'r
ef:manager', 'name:gv', 'name:gu', 'gps:vdop', 'name:uk', 'na
me:pt', 'surveillance:type', 'gps:hdop', 'name:fi', 'paymen
t:feathercoin', 'addr:full', 'name:ku', 'contact:website', 'n
ame:it', 'name:sk', 'name:botanical', 'payment:credit_cards',
'name:es', 'contact:email', 'alt_name:en', 'contact:phone',
'seamark:type', 'is_in:continent', 'disused:public_transpor
t', 'payment:bitcoin', 'addr:country', 'is_in:country', 'is_i
n:country_code', 'building:levels', 'payment:notes', 'name:p
l', 'disused:highway', 'payment:coins', 'internet_access:fe
e', 'name:hu', 'disused:railway', 'source:name', 'name:fr',
'name:ja', 'generator:method', 'historic:civilization', 'nam
e:vi', 'name:ru', 'tower:type', 'name:de', 'addr:housename',
'generator:source', 'addr:postcode', 'addr:city', 'payment:el
ectronic_purses', 'payment:telephone_cards', 'addr:housenumbe
r', 'addr:street', 'name:zh_pinyin', 'name:zh', 'name:en']
-----------------------------------------------------------
Strings containing duoble colons:
['service:bicycle:retail', 'service:bicycle:repair', 'servic
e:bicycle:pump', 'addr:street:en', 'seamark:mooring:categor
y', 'generator:output:electricity']
***********************************************************




***********************************************************
Problem 3: pinyin as names
First 100 tag element containing names:
[('name', u'\u5317\u4eac\u5e02'), ('name', u'\u6545\u5bab'),
('name', u'\u6728\u6a28\u5730\u5317'), ('name', u'\u96cd\u548
c\u5bab\u6865\u4e1c'), ('name', u'\u7389\u6843\u56ed'), ('nam
e', u'\u524d\u95e8'), ('name', u'\u94f8\u949f\u5382'), ('nam
e', u'\u5fb7\u5185\u7518\u6c34\u6865'), ('name', 'Guxiang 2
0'), ('name', u'\u5c0f\u5e84\u8def\u53e3\u897f'), ('name', 'T
he Bookworm'), ('name', u'\u5317\u4eac'), ('name', u'\u6c99\u
6ee9\u8def\u53e3\u897f'), ('name', u'\u949f\u697c'), ('name',
u'\u6984\u6746\u5e02'), ('name', u'\u5317\u4eac\u4e1c'), ('na
me', u'\u4e30\u53f0'), ('name', u'\u5317\u4eac\u5927\u5b66\u8
97f\u95e8'), ('name', u'\u6d77\u6dc0\u9ec4\u5e84\u5357'), ('n
ame', u'\u660c\u5e73\u533a'), ('name', u'\u671d\u9633\u533
a'), ('name', u'\u5d07\u6587\u533a'), ('name', u'\u5927\u538
2\u56de\u65cf\u81ea\u6cbb\u53bf'), ('name', u'\u5927\u5174\u5
33a'), ('name', u'\u4e1c\u57ce\u533a'), ('name', u'\u4e30\u53
f0\u533a'), ('name', u'\u623f\u5c71\u533a'), ('name', u'\u56f
a\u5b89\u53bf'), ('name', u'\u6d77\u6dc0\u533a'), ('name',
u'\u6000\u67d4\u533a'), ('name', u'\u5eca\u574a\u5e02'), ('na
me', u'\u95e8\u5934\u6c9f\u533a'), ('name', u'\u5bc6\u4e91\u5
3bf'), ('name', u'\u5e73\u8c37\u533a'), ('name', u'\u4e09\u6c
b3\u5e02'), ('name', u'\u987a\u4e49\u533a'), ('name', u'\u77f
3\u666f\u5c71\u533a'), ('name', u'\u901a\u5dde\u533a'), ('nam
e', u'\u897f\u57ce\u533a'), ('name', u'\u9999\u6cb3\u53bf'),
('name', u'\u6b66\u6e05\u533a'), ('name', u'\u5ba3\u6b66\u533
a'), ('name', u'\u6dbf\u5dde\u5e02'), ('name', u'\u5e7f\u963
3\u533a'), ('name', u'\u5b89\u6b21\u533a'), ('name', u'\u65b
0\u8857\u53e3\u8c41\u53e3'), ('name', u'\u6e05\u534e\u5927\u5
b66\u897f\u95e8'), ('name', u'\u5706\u660e\u56ed\u5357\u95e
8'), ('name', u'\u4e94\u9053\u53e3'), ('name', u'\u57ce\u94c
1\u67f3\u82b3\u7ad9'), ('name', u'\u67f3\u82b3'), ('name',
u'\u828d\u836f\u5c45'), ('name', u'\u5149\u7199\u95e8'), ('na
me', u'\u67f3\u6751\u7ebf\u8def\u6240'), ('name', u'\u82c7\u6
c9f'), ('name', u'\u4e0a\u6e05'), ('name', u'\u5317\u4eac\u7a
d9\u4e1c'), ('name', u'\u5317\u4eac\u7ad9\u897f'), ('name',
u'\u671b\u4eac\u897f'), ('name', u'\u4e1c\u76f4\u95e8'), ('na
me', u'\u897f\u76f4\u95e8'), ('name', u'\u5927\u949f\u5bfa'),
('name', u'\u77e5\u6625\u8def'), ('name', u'\u9f99\u6cfd'),
('name', u'\u970d\u8425'), ('name', u'\u7acb\u6c34\u6865'),
('name', u'\u9890\u548c\u56ed\u5317\u5bab\u95e8'), ('name',
u'\u7a7a\u519b\u6307\u6325\u5b66\u9662'), ('name', u'\u9890\u
548c\u56ed'), ('name', u'\u5927\u77f3\u6865\u5317'), ('name',
u'\u5c6f\u4f43\u6751'), ('name', u'\u6c38\u4e30\u5c6f\u897f\u
53e3'), ('name', u'\u6c38\u4e30\u4e61\u8f9b\u5e97'), ('name',
u'\u5730\u94c1\u751f\u547d\u79d1\u5b66\u56ed\u7ad9'), ('nam
e', u'\u516b\u8fbe\u5cad\u957f\u57ce'), ('name', u'\u8377\u82
b1\u5e02\u573a'), ('name', u'\u9890\u548c\u56ed\u5317\u5982\u
610f\u95e8'), ('name', u'\u5317\u4eac\u53e4\u89c2\u8c61\u53f
0'), ('name', 'Palace Hotel'), ('name', u'\u9f13\u697c'), ('n
ame', 'Five Dragon Pavilions'), ('name', u'\u5999\u5e94\u5bf
a'), ('name', 'Circular City'), ('name', u'\u9759\u5fc3\u658
b'), ('name', 'Heavenly King Hall'), ('name', u'\u4e5d\u9f9
9\u58c1 (Nine Dragon Screen)'), ('name', 'Long Corridor'),
('name', u'\u6e05\u664f\u822b'), ('name', u'\u4ec1\u5bff\u6bb
f'), ('name', u'\u4e07\u82b1\u9635'), ('name', u'\u5927\u6c3
4\u6cd5'), ('name', u'\u5eb7\u5e84\u7ad9'), ('name', u'\u516
b\u8fbe\u5cad'), ('name', u'\u826f\u4e61'), ('name', u'\u4e5
d\u9f99\u58c1'), ('name', 'China Glacier Museum'), ('name',
u'\u6cd5\u6d77\u5bfa'), ('name', u'\u7fe0\u5fae\u5c71'), ('na
me', u'\u6302\u7532\u5854'), ('name', u'\u4e09\u5c94\u53e3')]
************************************************************
```

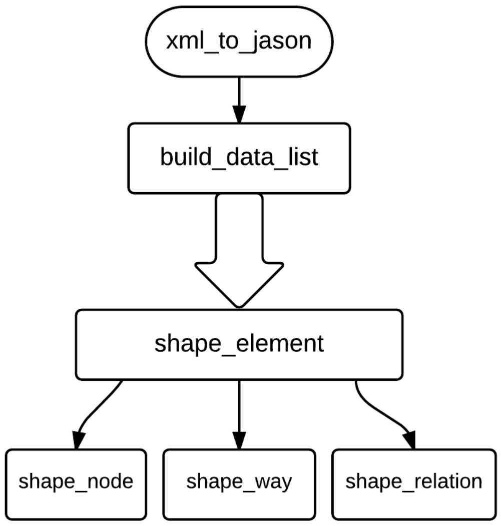
1.2 Three Problems

Three problems are presented:

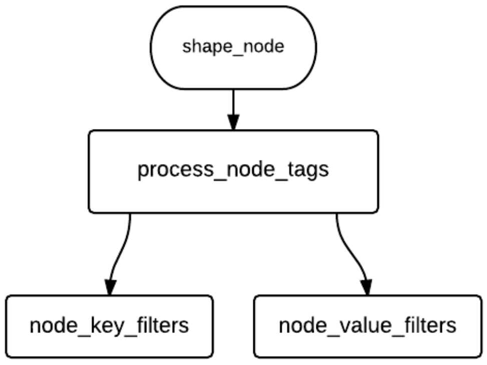
1. Problems associated with different type of keys of tag elements. Uniformity of keys makes search through Mongodb more easily. There are three procedures I have done to improve uniformity of keys. First, I find a handful chinese characters as keys in the xml. And I will discard these tag. Second, many keys with multiple words containing underscore("_") or hyphen("-"). I replace all the hyphens with underscore for uniformity of keys. Third, all the uppercase letters are replaced by lower case letters. These procedures are done in "xml_to_json.py".
2. Problem with colons in keys of tag element. As we can see from the result of running "audit.py". Some keys contain a single colon(":") and some keys contain double colons. Usually, colon means existence of sub-class. However, a smaller number of keys containing double colons which indicate level-two sub-class. I think level-one subclass is a standard way of dealing with these keys. I replace the second colon with underscore. And all these keys will only contain one colon. I construct sub-classes from these keys. For example, I have three keys with colon: "name:ch:simplified", "name:ch:traditional" and "name:en". These keys will form such structure: name_other:{ch_simplified:..., ch_traditional:..., en:...}. Noticing that I add "_other" suffix to make a new group of word. The procedure is also in "xml_to_json.py".
3. Problem of names. I noticed that some values of 'name' tag are neither Chinese character nor English words, such as "Guxiang 20" in the above output of "audit.py". These names are all pinyin which is a phonetic system of chinese characters. And these names are hardly used as names in Chinese. Luckily, these pinyins can be easily translated into more meaningful Chinese words. I replace all the pinyin in 'name' tag in the process. The procedure are also in "xml_to_json.py".

1.3 Structures of Cleaning Process

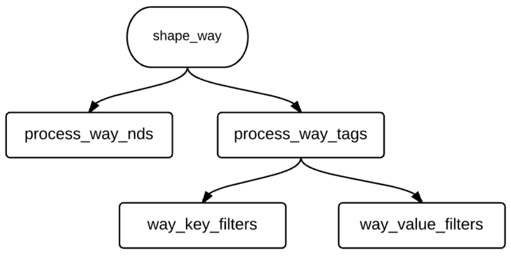
I will briefly introduce my structures of program which dealing with above problems. The following figure show the main structures of "xml_to_json.py".



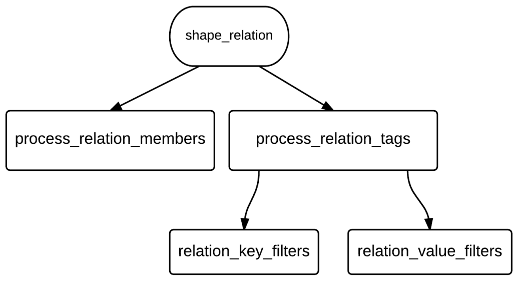
The program will build a list containing all the dictionaries from xml. The building process of this huge list is done by shaping individual element. There are three kinds of shaping method corresponding to three kinds of elements which is node, way and relation. The structure of shaping node is as following:



and way:



and relation:



In the function "process_node_tags", "process_way_tags" and "process_relation_tags", problem 2 and 3 are actually solved. Problem 1 are solved in "node_key_filters", "way_key_filters" and "relation_key_filters". The programs are structured such way to accommodate more possible data wranglings in future.

2. Data Overview

Data shaped by above process is imported into Mongodb. The json objects are insert into database "example" as collection "beijing_maps".

In [2]:

```
from db_functions import *
insert_maps(JSON_FILE, DB_NAME)
all_collections = get_collections(DB_NAME)
print JSON_FILE
print DB_NAME
```

```
../beijing_china.osm.json
examples
```

2.1 Size of xml file and json file

In [3]:

```
!du -h ../beijing_china.osm
!du -h ../beijing_china.osm.json
```

```
130M    ../beijing_china.osm
142M    ../beijing_china.osm.json
```

2.2 Some Total Counts

2.2.1 Total count of all the documents.

In [4]:

```
all_collections.find().count()
```

Out[4]:

```
699609
```

2.2.2 Total count of nodes, ways and relations.

In [5]:

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

Out[5]:

```
605204
```

In [6]:

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

Out[6]:

```
88958
```

In [7]:

```
all_collections.find({"type":"relation"}).count()
```

Out[7]:

```
5447
```

2.2.3 Counts associated with 'tag' element

Number of node, way and relation elements containing 'tag'.

In [8]:

```
all_collections.find({"type":"node", "tag":  {"$exists": 1}}).count()
```

Out[8]:

```
34490
```

In [9]:

```
all_collections.find({"type":"way", "tag":  {"$exists": 1}}).count()
```

Out[9]:

```
88240
```

In [10]:

```
all_collections.find({"type":"relation", "tag":  {"$exists": 1}}).count()
```

Out[10]:

```
5437
```

Number of node, way and relation elements containing 'name'.

In [11]:

```
all_collections.find({"type":"node", "tag.name":  {"$exists": 1}}).count()
```

Out[11]:

```
9610
```

In [12]:

```
all_collections.find({"type":"way", "tag.name":  {"$exists": 1}}).count()
```

Out[12]:

```
17124
```

In [13]:

```
all_collections.find({"type":"relation", "tag.name":  {"$exists": 1}}).cou
nt()
```

Out[13]:

```
4761
```

2.3 Some knowledge of updating time

Earliest time for updating of the map.

In [14]:

```
sorted_times = get_unique_time_sorted(all_collections)
print sorted_times[0]
```

```
2007-03-14 18:09:10
```

Latest time when the map file is downloading.

In [15]:

```
print sorted_times[-1]
```

```
2015-08-28 14:04:24
```

The time when people update most frequently on this map.

In [16]:

```
pipeline = [{'$group':{'_id':'$timestamp',
                       'count':{'$sum':1}}},
            {'$sort':{'count':-1}},
            {'$limit':1}
            ]
list(all_collections.aggregate(pipeline))
```

Out[16]:

```
[{u'_id': u'2015-06-12T06:31:41Z', u'count': 92}]
```

3. Additional Ideas

3.1 Incompleteness and Inconsistency

The map is far from finishing. There are many entities missing. More data is needed to make the map useful. Too few independent nodes are included in the map. There are total 605204 nodes in the map. Only 9610 have a name. Let us see how many nodes are used in defining a way instead of independent entities.

In [17]:

```
pipeline = [{'$match':{'type':'way'}},
            {'$unwind' : '$nodes'},
            {'$group' : {'_id':'$nodes'}}
       ]
print 'The number of nodes used in  defining ways is ' + str(len(list(aggr
egate(all_collections, pipeline)))) + '.'
print 'Total number of nodes is ' + str(all_collections.find({'type':'nod
e'}).count()) + '.'
print 'Number of independent nodes is ' + str(all_collections.find({'typ
e':'node'}).count() - \
                                              len(list(aggregate(all_colle
ctions, pipeline)))) + '.'
```

```
The number of nodes used in  defining ways is 596448.
Total number of nodes is 605204.
Number of independent nodes is 8756.
```

Apparently, 8756 nodes is highly insufficient describing a huge city like Beijing. There are total 17124 named ways in this may, I believe this would roughly enough. Based on this conclusion, I suggest that collecting information of nodes(entities) on this map is priority.

The names of nodes and ways are very important. The names in the map of Beijing should be in Chinese character instead of pinyin which is the official phonetic system of the Mandarin. Pinyin is meaningless in writing or printing the Mandarin. However, I find many names of pinyin in the map. In other words the names are not consistent.

In [18]:

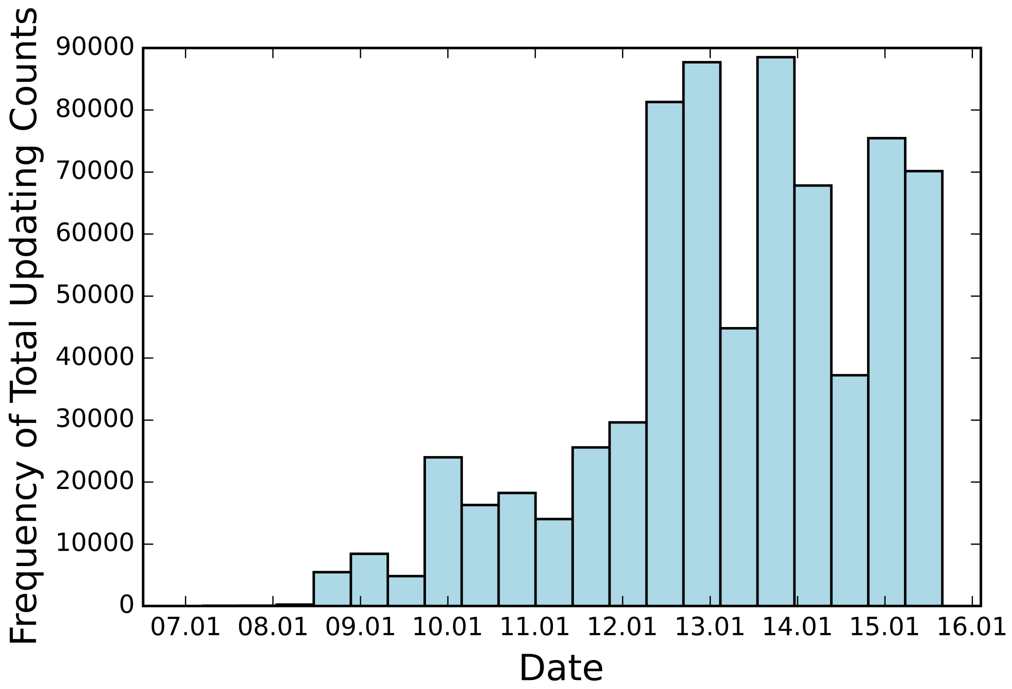
```
from process_util import *
result = read_word_list(NODE_NAME_FILE)
print 'Total number of names of nodes is '+ str(len(result)) + '.'
print 'Number of pinyin is ' + str(count_map(pinyin_reg, result)) + '.'

result = read_word_list(WAY_NAME_FILE)
print 'Total number of names of ways is ' + str(len(result)) + '.'
print 'Number of pinyin is ' + str(count_map(pinyin_reg, result)) + '.'
```

```
Total number of names of nodes is 5822.
Number of pinyin is 988.
Total number of names of ways is 7724.
Number of pinyin is 721.
```

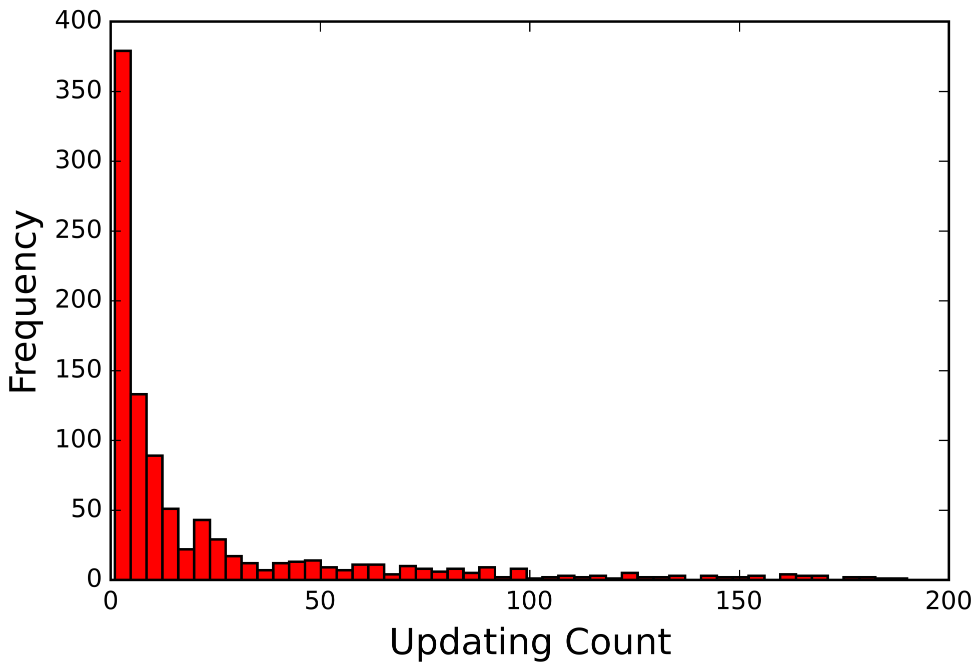
Based on the abservation, I recommend ones who make changes to the names to use local language for consistency.

3.2 Commit history of the map.

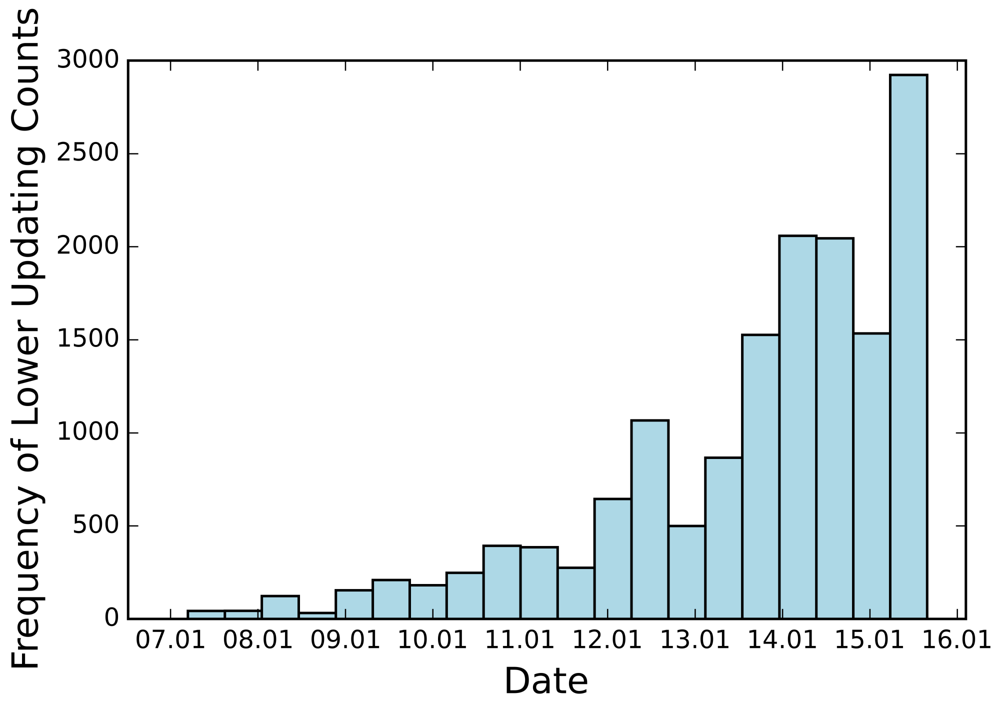
First, let's see the varying of total numbers of updates according to dates. The updating counts keeps growing before year 2012. However, after a sudden increasing in 2012, the growing stopped. The image is produced by script in db_functions.py using Mongodb query and matplotlib.



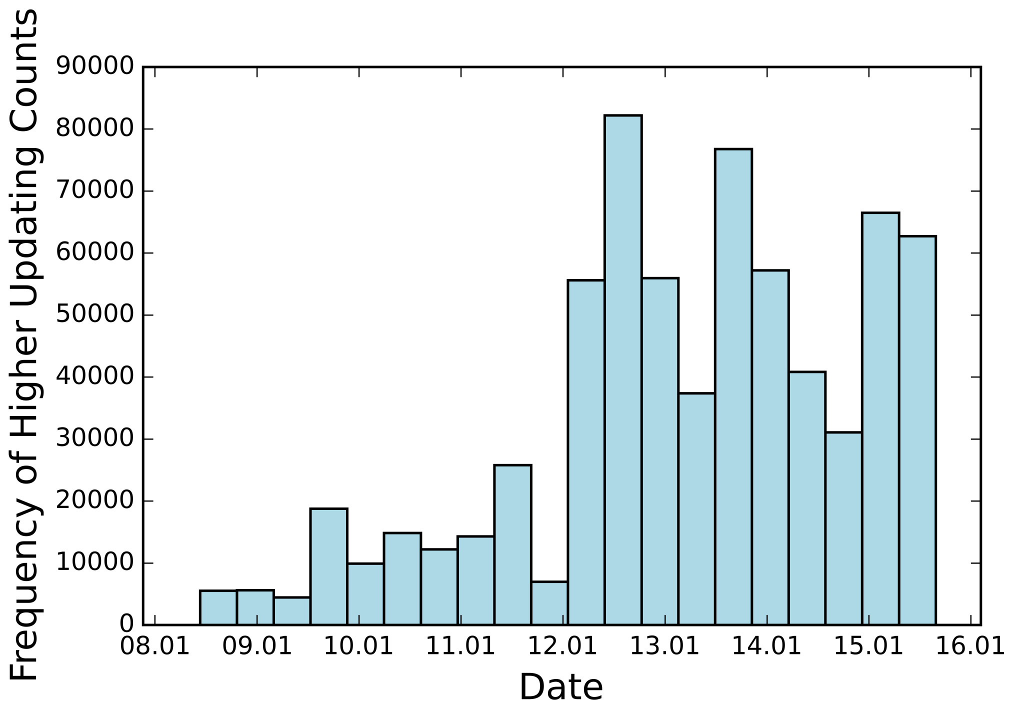
One may ask, why does the counting of updating grow like this way. Did the number of users of Openstreetmap stopped to increase since 2012? Let us first look at the distribution of updating counts for different users. The distribution is highly skewed. Most users only updating less a few times. And most of updates are commited by a handful users. Total number of users is 1102. And almost 90% user updating less than 100 times.The image is produced by script in db_functions.py using Mongodb query and matplotlib.



Majority of users updating less than 100 times. I will call this part of users "Lower updating users". The rest of user is "higher updating users". Since most users are lower updating users. I will plot how contribution of lower updating users evolved acrossing time. Following is the figure. The figure shows that during last 8 years, the counting is consistently increasing. I believe that this is the evident which shows the Openstreetmap is making significant progress during last few years. The image is produced by script in db_functions.py using Mongodb query and matplotlib.



But how the sudden change of updating counts happened in year 2012? The plot of higher updating users reveals the answer. The plot is almost identical to the total counting trend. Because of the participation of some higher updating users in year 2012 the counting is increasing dramatically at that time point. The image is produced by script in db_functions.py using Mongodb query and matplotlib.



4. Conclusion

The map data of Beijing from Openstreetmap is carefully examined and summarized. Important features such as scale, user, updating time and other statistical numbers were given by this article. Several problems associated with this dataset are proposed and most of them are solved in a systemic way. These problems include inconsistency of keys, irregularity of words and improper names of entities. I observed the map data is far from completeness. The map lack a lot of descriptions of nodes especially. And I also noticed that many names in the data set is in pinyin which is not a right way to identify map entities. Some commit history is examined and showed to verify the fact that the user of Openstreetmap is in a consistent growing.