OpenStreetMap Thinking In Beiing with MongoDB

*Tina Wang*

Map Area: Beijing China

[*https://www.openstreetmap.org/relation/912940*](https://www.openstreetmap.org/relation/912940)

# Author Says

Just begin the analysis, I must thank to Udacity, without this deadline I absolutely will give up writing this paper. There is a singer in China, he has a song “One Night in Beijing” which I like very much. So here I want to make my so-called paper as “OpenStreetMap Thinking in Beijing”, I have to say my analysis is not good, robot can make more beautiful and professional analysis, but I can feel more happiness than it does.

Ok, return back to business, I choose Beijing, there is no big reason, just because it’s my second hometown, I have lived here for twelve years. So I want to do something for it, at the beginning I need analysis its map ☺

# Overview

## MongoDB

I will parse OSM file into a json file before importing into MongoDB.

### Design

I name the db as ‘udacity’, and the collection as ‘map’, this collection contains both <node> and <way>,

**Parsing Rules**

* For <node> element, type=’node’; for <way> element, type=’way’
* Created

All attributes for <node> and <way> in array [ **"version"**, **"changeset"**, **"timestamp"**, **"user"**, **"uid"**] are included in “created“:{}

* Pos

Both <node> and <way>, if it contains lat and lon attribute, they will go to “pos”:[lat,lon]

* Address

Both <node> and <way> contain some <tag> nodes, for those tags whose k attribute value begins with ‘attr:’ and only contain 2 ‘:’ will go to “address”:{}

* Name attribute for <node> and <way> if have

name attribute will go to “name”:{“\_def”:””}

and if we

and if there is translation <tag> node for name, it will also go to “name”:{}, eg

<tag k=”name:de” v=”XXXX”>

It will go to

“name”:{“\_def”:”…”,”de”:”XXXX”}

* For other attributes of <node> and <way>

I will use the same dispose as “name”

### Sample

**Node**

|  |
| --- |
| {  "type": "node",  "created": {  "version": "90",  "timestamp": "2017-11-05T09:06:50Z",  "changeset": "53521955",  "uid": "554973",  "user": "dgitto"  },  "id": "25248662",  "pos": [  39.9059631,  116.391248  ],  "admin\_level": "2",  "alt\_name": {  "\_def": "\u5317\u4eac",  "en": "Peking",  "eo": "Bej\u011dino",  "fr": "Beijing",  "is": "Peking",  "sv": "Peking"  },  "capital": "yes",  "name": {  "\_def": "\u5317\u4eac\u5e02",  …  "ja": "\u5317\u4eac\u5e02",  "zh": "\u5317\u4eac\u5e02"  },  "old\_name": {  "\_def": "\u5317\u5e73",  "vi": "B\u1eafc B\u00ecnh"  },  "place": "city",  "population": "13133000",  "rank": "0",  "ref": {  "\_def": "\u4eac",  "en": "BJ",  "vi": "Kinh"  },  "wikidata": "Q956",  "wikipedia": "en:Beijing"  } |

**Way**

|  |
| --- |
| {  "type": "way",  "created": {  "version": "19",  "timestamp": "2016-12-06T17:34:07Z",  "changeset": "44215196",  "uid": "3450290",  "user": "ff5722"  },  "id": "4231222",  "name": {  "\_def": "\u5e7f\u573a\u897f\u4fa7\u8def",  "en": "West Guangchang Road",  "zh": "\u5e7f\u573a\u897f\u4fa7\u8def"  },  "oneway": "yes",  "highway": "primary",  "node\_refs": [  "25248785",  "330841128",  "2347911578",  "25248788",  "1891268439",  "2347911594",  "342819253",  "25248790"  ]  } |

## Flask Online Demo

For readers to get a direct feeling of my project, I provide an online site and some rest services for you to review

### Home page

<http://xingboedu.com:6000>

### Rest Services

<http://xingboedu.com:6000/invalidPostalCode> (GET)

[http://xingboedu.com:6000/**topusers/<int:cnt**](http://xingboedu.com:6000/topusers/%3cint:cnt)**> (GET)**

# Issues in the Map

I find next issues in the map, and all updates are on “mapp” collection, not on source “map” collection.

### Postal Code/Adcode

Postal codes in Beijing are all 6 digits which begin with ‘10’. So I do a query to find all invalid codes

db.mapp.find({"address.postcode":{$exists:true,$not:/10\d{4}/}})



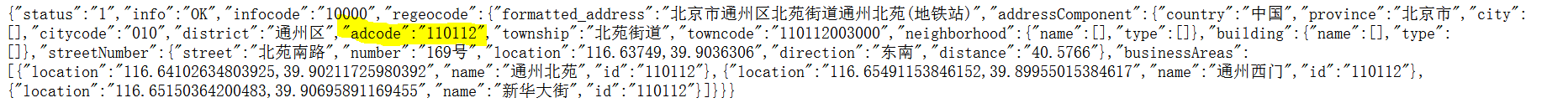
From the result, we can see there are not many invalid postcode issues here.

Then next we need try to find a way to correct them or remove them if cannot find.

I know some corps provide APIs to get information by location( [longitude and latitude](http://www.baidu.com/link?url=3lZGqdpdSFDFihI8lpLMOb5qlgKG6hyR3lgSaKn_v7yarMDMkMr9oeD_Mf0JUOFLmCSKJU8tYUOOhJmQOUUIcND1SHy3AWLeCaDqTnDDnlnnDZh4p1yTJW476TFjbJTk)). Here I tried APIs from amap:

<http://restapi.amap.com/v3/geocode/regeo?output=json&location=116.6372277,39.9039351&key=5207eb75880ec13ec118ef705b0dfea6>

The response is as below:



There is a ‘adcode’ here, unfortunately this adcode is not postcode. Most popular map Corps provide adcode (not postcode). So I decide to enrich our data to provide adcode.

I call **for** one **in** map.find({**"pos"**:{**"$exists"**:**True**}}): to loop through all rows which have **pos** defined. After running a while, I found it exited with an error:

pymongo.errors.CursorNotFound: Cursor not found

Googling for this error, get it that the for … find() takes too long, the cursor is closed automatically, here I add *no\_cursor\_timeout = True* when call find on the collection, and we need close the cursor explicitly after for…

This time, it will finish until all rows are disposed. But I noticed most rows’ adcode is “”, then I tried to print amap rest service’s json response, found it has exceeded the daily calling count, I modify the code to check the status code in the json response: if found status==0, will exit without wasting time to call.

Note: Please see addAdcode() in clean.py for the complete method.

Although adcode is added, I still want to try to fix the postcode errors we found. Then I baidu (Baidu is a popular searching engine of China), find next wet service:

**http://www.webxml.com.cn/WebServices/ChinaZipSearchWebService.asmx/getZipCodeByAddress?theProvinceName=&theCityName=&theAddress=&userID=**

After play manually for a while I found it’s not so good that its fussy searching does NOT work well, many times it cannot return a single result. But it’s the only public API I can use now, so I still plan to try it in the code.

After run, I found it only find out one zipcode, sounds not good not bad.

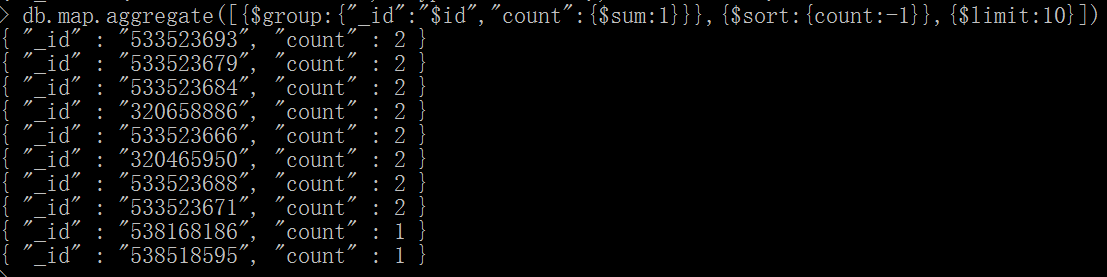
Note: please see fetchPostalCode() in clean.py for the complete method

Then when I browse the left 10 rows, I found street in one row is in English language(not in Chinese), and if I translate it into Chinese and search manually using the rest service, it cannot find at the first try, then I tried not to use the whole string, whereas just use a keyword, then search, finally find a postal code out. So I suppose what I need do next is to make the manual process automated: we need a translation public rest API and a splitting word open source which is similar like the one in Elastic Search.

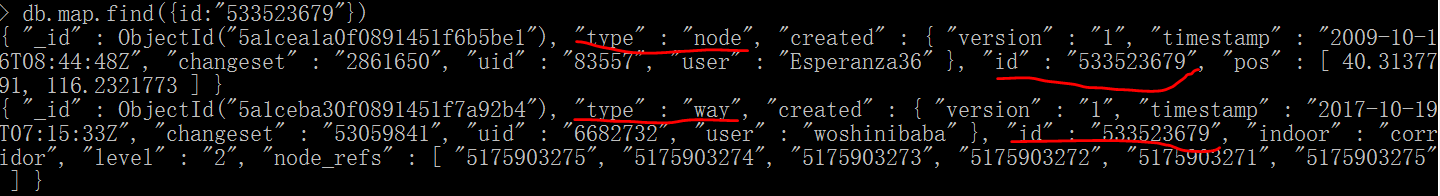
### Duplication Elements

At first, I want to confirm whether there are duplicated IDs in map, I did next query:

db.map.aggregate([{$group:{"\_id":"$id","count":{$sum:1}}},{$sort:{count:-1}},{$limit:10}])



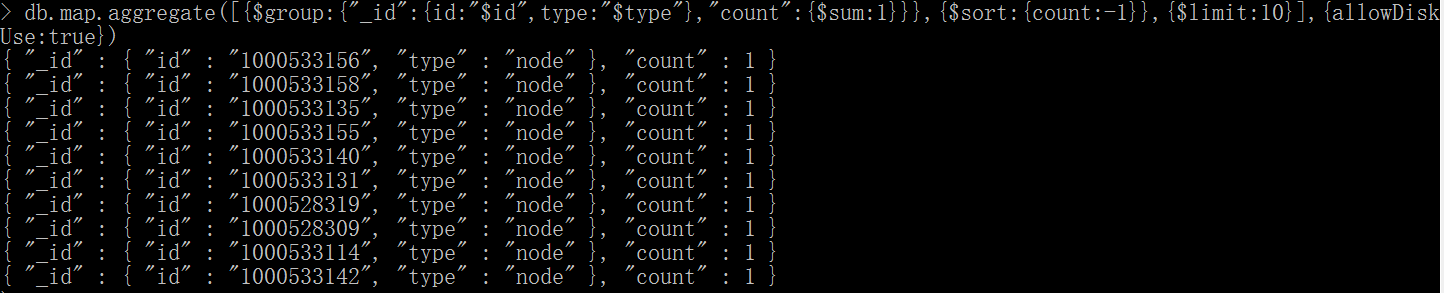
There are some 8 duplication, then I query one duplication as below:



Obviously, it means a node and a way can use same id. **Is it valid?**

I would like to know whether there is any duplicated node or duplicated way, this time I use id and type is composite id:

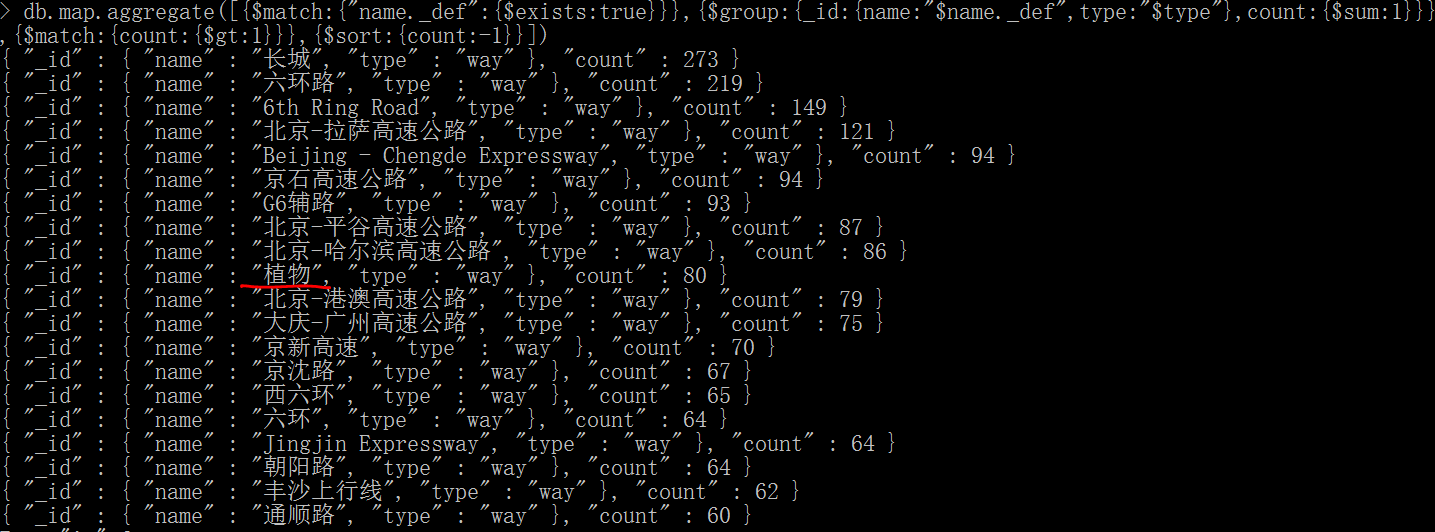
db.map.aggregate([{$group:{"\_id":{id:"$id",type:"$type"},"count":{$sum:1}}},{$sort:{count:-1}},{$limit:10}],{allowDiskUse:true})



OK, there is no duplication.

Next, I want to check whether there is any duplicated name in node and way respectively, I use name.\_def and type as composite ID:

db.map.aggregate([{$match:{"name.\_def":{$exists:true}}},{$group:{\_id:{name:"$name.\_def",type:"$type"},count:{$sum:1}}},{$match:{count:{$gt:1}}},{$sort:{count:-1}}]).



There is a “植物” row, it means not a way, let me query it now

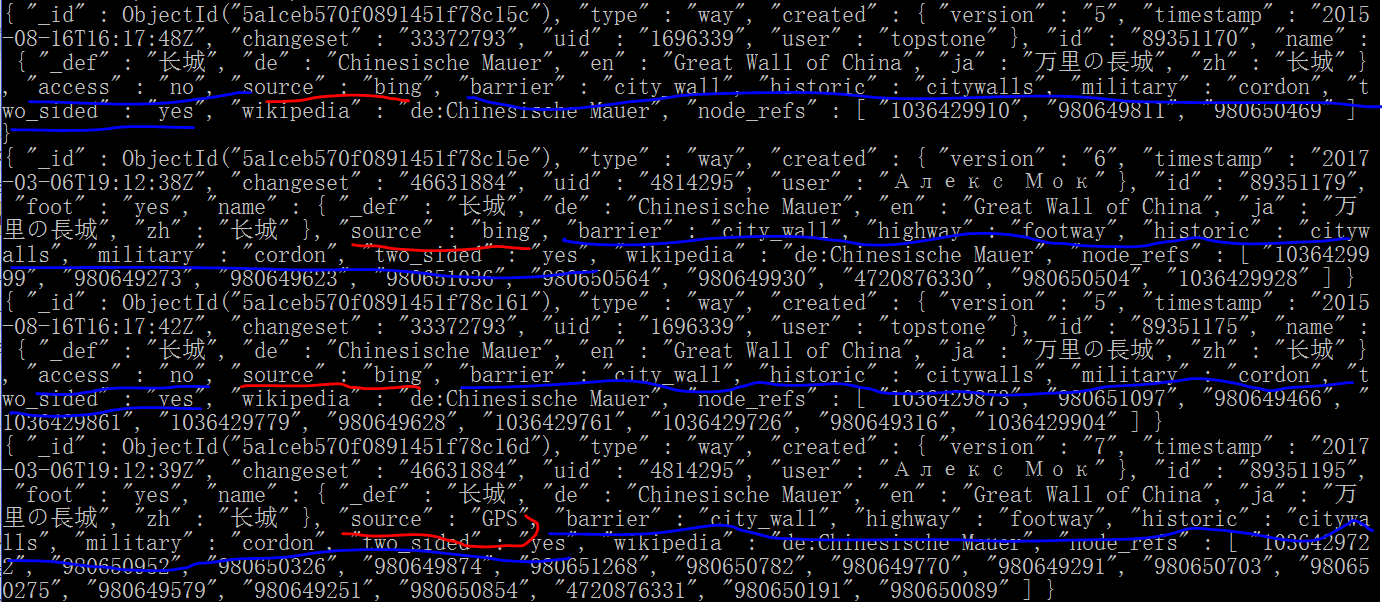
db.map.find({"name.\_def":"植物"})



We can see these records are only different in height and node\_refs.

I suppose this’s a scenario of duplication

Next, I manually queried name.\_def=”长城”:



From the sample output, we can also manually judge they are duplicated ones: although they are from different user and source and they have different node\_refs, they share some same valued fields and these fields are key fields to identify duplication.

Here, I am a little headache, different duplications has their special scenarios, for example “植物” and “长城”,they use different fields to judge duplication. So I suppose a simple aggregation or mapReduce cannot handle all of them.

A knowledge base might be set up to handle them.

Eg: for “长城” records

We have a base about it

key is “长城”, once a record come it, the system analyze it, what fields it contain and their values, when a new record come, the system analyze it, whether it has new field not recorded before and its value, then judge whether it’s a duplicated one.

I just leave a blank cleanDuplication method in clean.py, I will try to finish it before the class finish.

### “Source” Clean

When I dispose duplications in previous section, source attribute comes into my eyes, because it’s an important field when identify unique rows.

Then I did next query:

db.map.distinct(“source”) 

From the result, we can see there are some sources are really same source, for example what I colored in the images.

Next I will clean some of them.

#### GPS

I use next query to find GPS variations:

db.map.distinct("source",{source:{$exists:true,$regex:/^gps/i}})



The blue colored ‘bing:GPS’ will be dispose as ‘Bing’ in next section.

Then I use next update command to clean:

db.mapp.update(

{source:{$exists:true,

$in:[

/^gps/i

]

}},

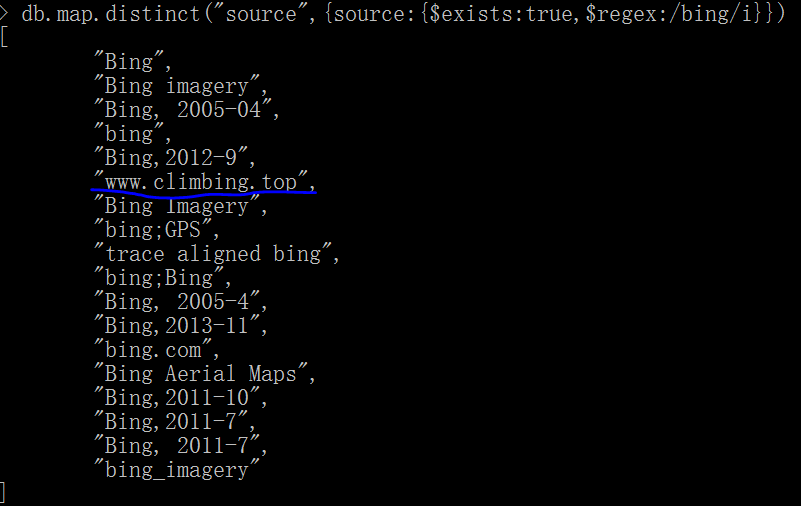
{$set:{source:'GPS'}},

{update:true,multi:true}

)

#### Bing

I did a query on how many Bing variation there are:



We can see the blue colored value is not a valid variation, so don’t clean it.

I will use next update command to update all Bing variations to ‘Bing’

db.mapp.update(

{source:{$exists:true,

$in:[

/^bing$/,

/^bing[\s,.\-;\_]/I,

/\sbing$/i

]

}},

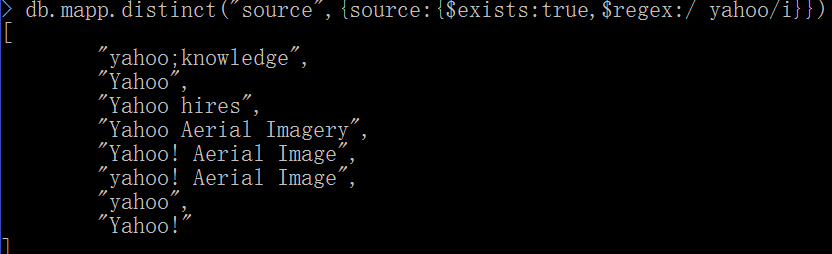
{$set:{source:'Bing'}},

{update:true,multi:true}

)

#### Yahoo

I did a query on how many Yahoo variation there are:



Then I use next update command to clean:

db.mapp.update(

{source:{$exists:true,

$in:[

/^yahoo$/i,

/^yahoo[\s;!]/I

]

}},

{$set:{source:'Yahoo'}},

{update:true,multi:true}

)

The complete code can be found in cleanSource method of clean.py

### English Name

As I said in last section, I found some rows’ name.\_def field contains non-Chinese characters,(many of them are English), whereas most rows’ name.\_def is Chinese string (English name is saved in name.en field). So I suppose I should clean the data to make all the rows more consistent in behavior.

There are some public translating rest services now, although there some daily free-charging calling max limitation, they can be used here.

I choose [**http://api.fanyi.baidu.com/api/trans/vip/translate**](http://api.fanyi.baidu.com/api/trans/vip/translate)**, which need register as developer at first**

When I write this paper, I have not figured out all my rules completely, next I just give a simple example:



The rule is that: once find name.\_def contains a keyword in [‘street’, ’road’, ‘expressway’], script will call the rest api to translate the whole string, once find a translation it will put original name.\_def in name.\_old, then replace name.\_def with the translation.

When run the code, I print out these original names, find some of them contain Chinese+Pinyin+English, the Chinese part is the real name, the Pinyin+English part is noisy and don’t need to translate because even translated, the translation of it is the Chinese part. For Example：

"万红西路 Wàn Hóng Xī Road"

“Wàn Hóng Xī” is PinYin of “万红西”, “路” is the zh translation of “road”

What we need do here is truncate name.\_def value directly as "万红西路”。

So I define a regex expression for PinYin, if find the name contains pinyin, the name will be truncated to the first space(There is no space between every word in Chinese). I know the rule is a little simple for now, and we may adjust it further.

Note: the method is translateEnglishName() in clean.py

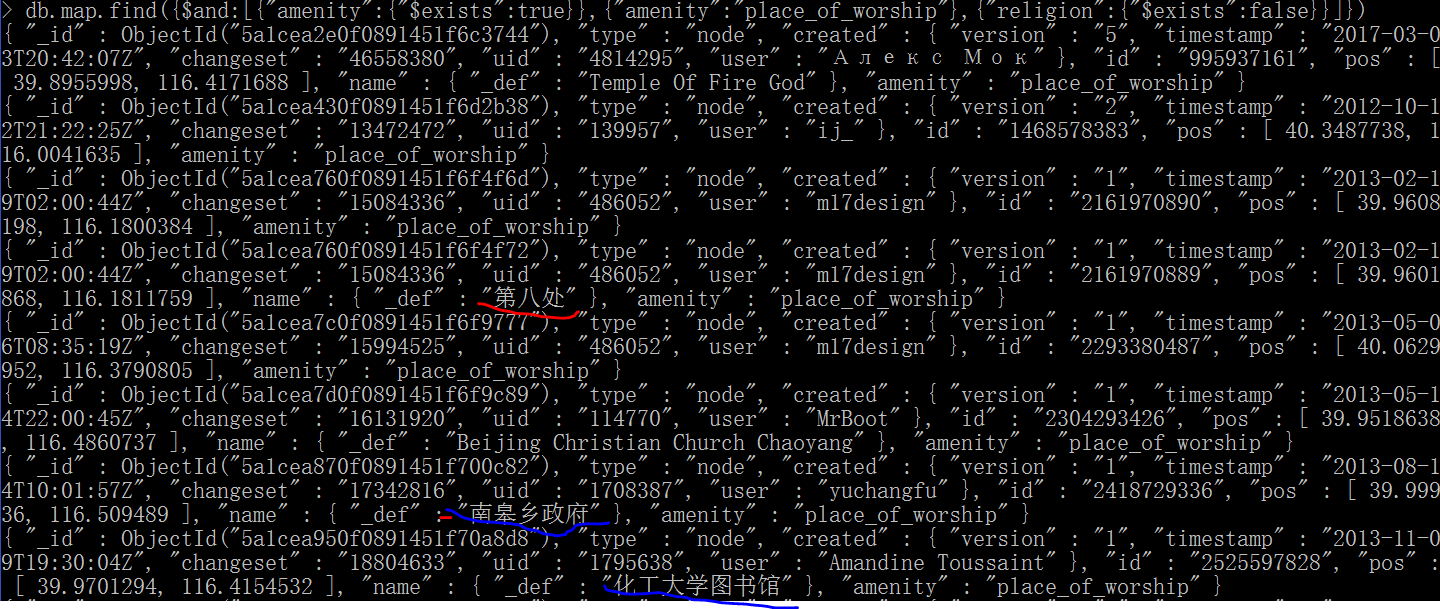
### Missing Data

#### Religion

Based on [Top N Religions](#_Top_N_Religions), I found there are some worship places missing religion field, so I run

db.map.find({$and:[{"amenity":{"$exists":true}},{"amenity":"place\_of\_worship"},{"religion":{"$exists":false}}]})

Next is the rows in first page

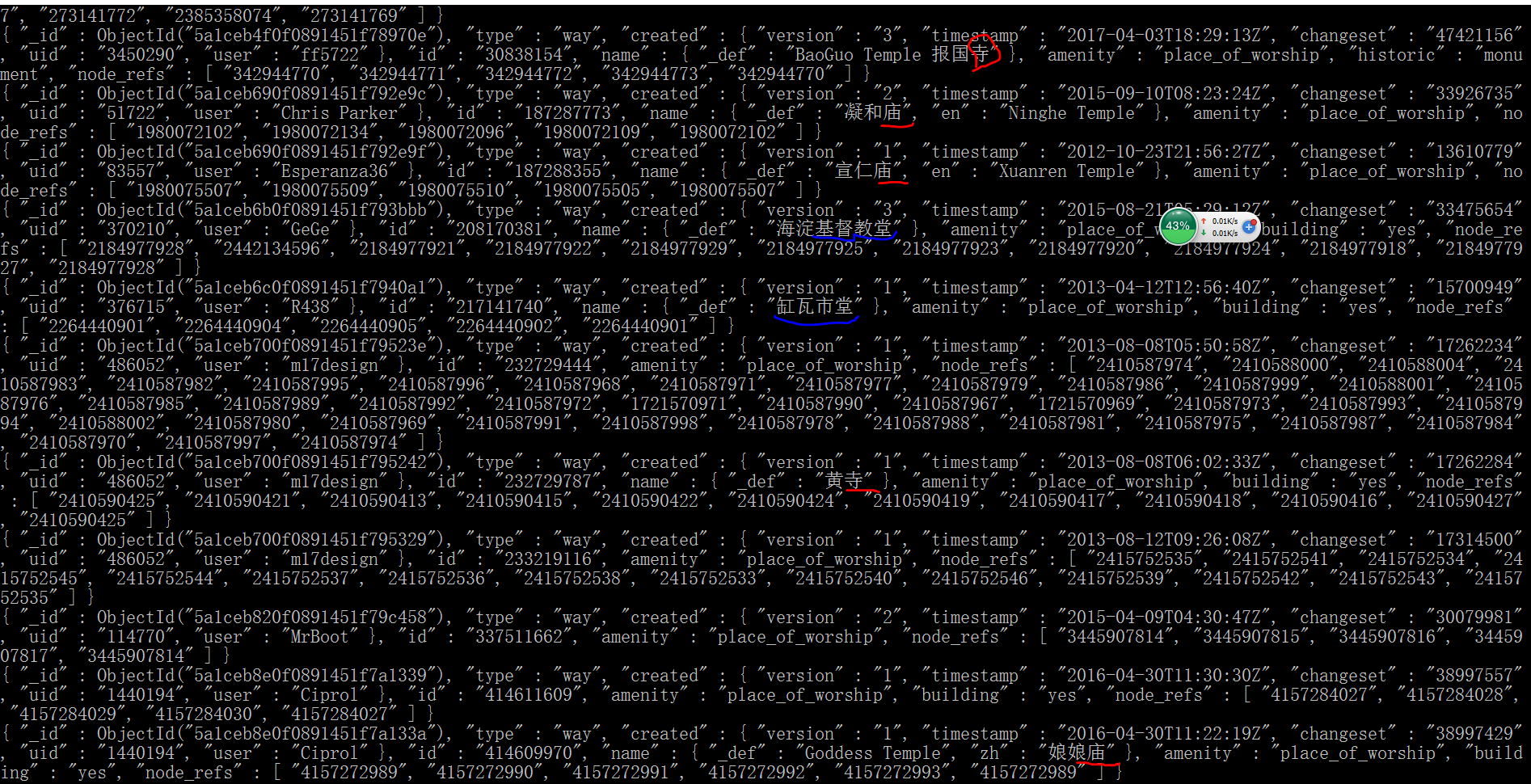


The red highlighted row use wrong name, Nearly all person in Beijing know it should be “八大处”，not “第八处”. But, I cannot see any robot can know it for now, I can know recognize it by eyes.

The blue highlighted rows are not worship places, I find a rule from them , if name contains “政府” or “图书馆”, they are not worship places, need correct.

Also, if a name contain ‘Christian’, its religion should be ‘Christian’, if contains ‘temple’, its religion should be ‘Buddhist’.

Then from next page, I find out more rules



The complete clean code is cleanReligion method in clean.py

#### Cuisine

Based on the [Top N cuisines](#_Top_N_cuisines) , I found some restaurants missing cuisine, then I did next query:

db.map.find({$and:[{"amenity":{"$exists":true}},{"amenity":"restaurant"},{"cuisine":{"$exists":false}}]})



I find it’s easy to judge a restaurant’s cuisine from its name manually, whereas not by robot if no knowledge base is available.

I tried to find a public rest service, not found for now. So I suppose I need to setup some simple rules at first to do a simple clean for now:

* If name.\_def or name.zh contains an item in array[‘酒家’,’酒楼’,’火锅’,’海鲜’],then the cuisine is ‘chinese’
* If name.\_def or name.en contains an item in array [‘india’], or name.\_def or name.zh contains an item in array[‘印度’], then the cuisine is ‘india’

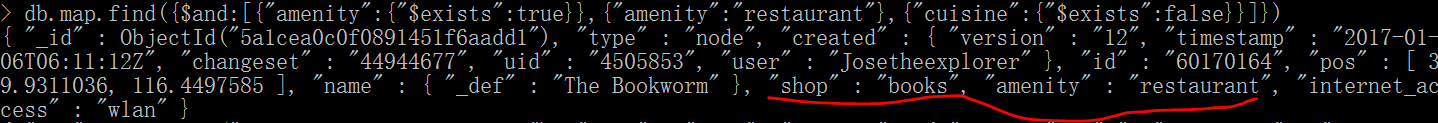
These 2 rules are very simple and are just a sample, we can make rules much more and more precise.

When run the code, find there are some extra keywords I did not find out when design the rules, so add rules.

The complete code is in cleanCuisine method of clean.py

Here, there is another special row when I browse data, See next example

This is a restaurant, but “shop”:”books”, so it’s a bookstore which sell meal?



Then I did a searching, find it’s a popular book bar where people can read book, drink coffee and eat fast food.

Bingo! It’s not an issue row.

### Cleaning Sequence

From the issues I mentioned in previous sections, we can find the analysis process might last several rounds, an previous mentioned issue may depend on the output of a latter found issue. Eg: Duplication Clean depends on Source Clean

So the final method running process of clean.py is next:

Missing Data

Postal code/Adcode

English Name

Source Clean

Duplication

# Map Overview

This section contains basic statistics about the dataset and the MongoDB queries used to do analysis.

All statistics are based on original map collection, not based on cleaned collection (map) because I did a duplication removal on it.

### File sizes

beijing\_china.osm ......... 192 MB

beijing\_china.osm.json .... 285 MB

### Number of documents

db.map.find().count()



### Number of nodes

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



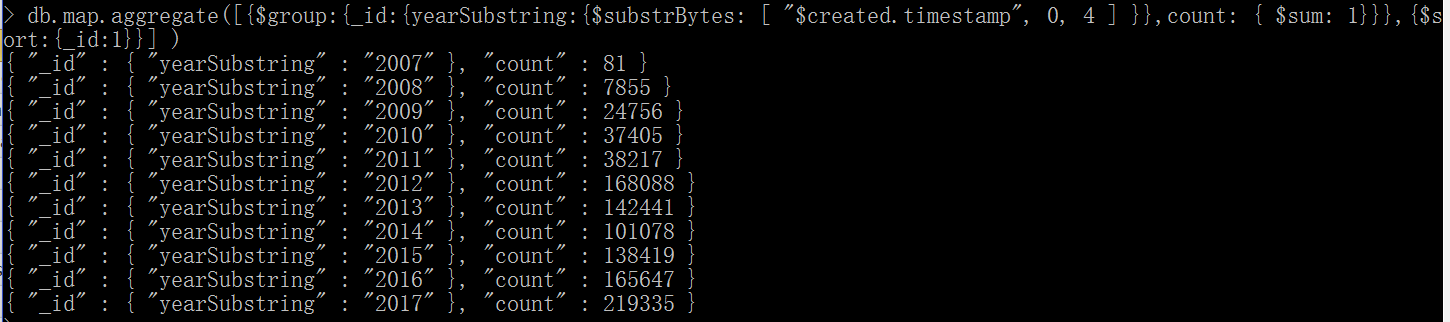
### Number of ways

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

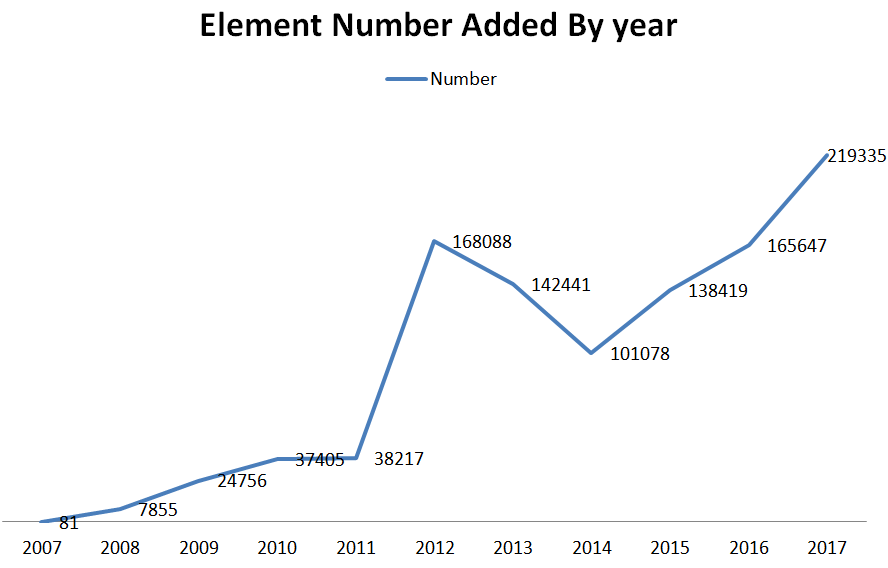


### Element Number added every year

> db.map.aggregate([{$group:{\_id:{yearSubstring:{$substrBytes: [ "$created.timestamp", 0, 4 ] }},count: { $sum: 1}}},{$sort:{\_id:1}}] )

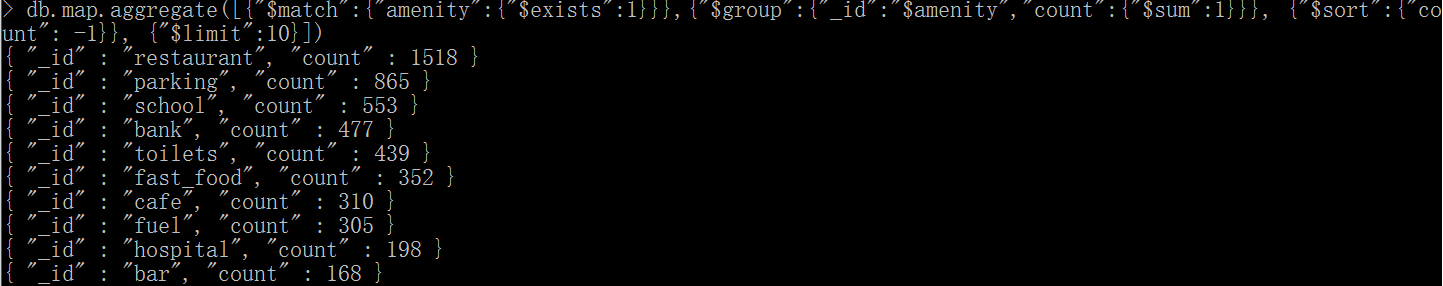


**I make a chart for the result**：



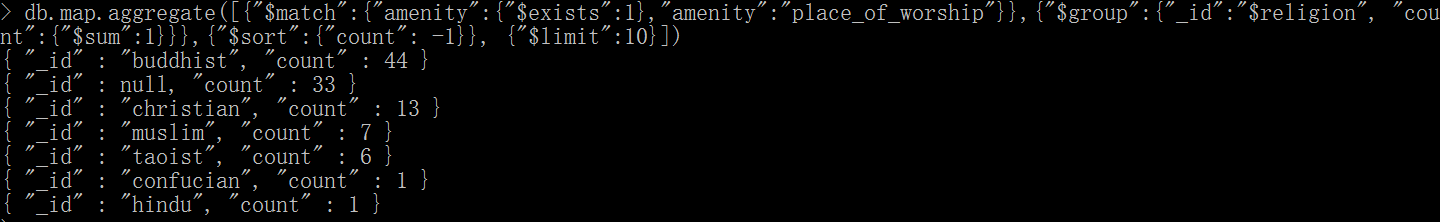
### Top N appearing amenities

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



### Top N Religions

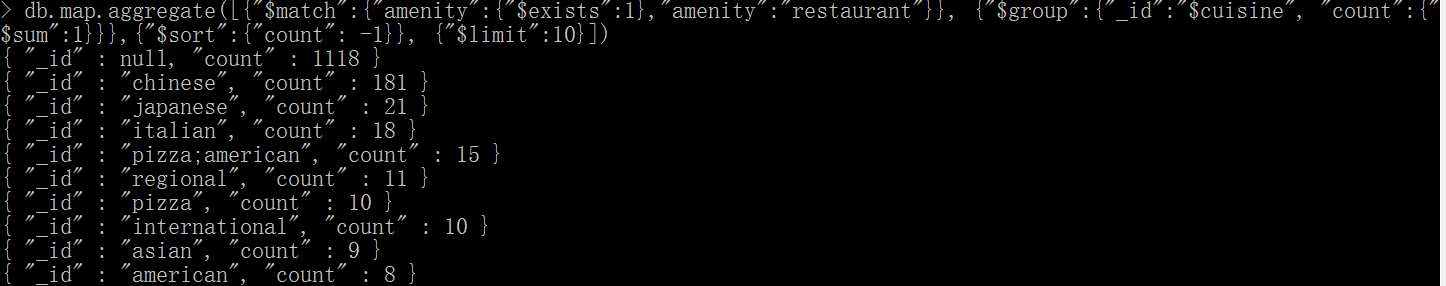
db.map.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"place\_of\_worship"}},{"$group":{"\_id":"$religion", "count":{"$sum":1}}},{"$sort":{"count": -1}}, {"$limit":n}])



Of course, In china Buddhist is the most popular religion, but here I noticed there is a “\_id”:null, which means missing religion field, so comes the issue I mentioned in the chapter “Issues in the Map” chapter.

### Top N cuisines

db.map.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"restaurant"}}, {"$group":{"\_id":"$cuisine", "count":{"$sum":1}}},{"$sort":{"count": -1}}, {"$limit":n}])



From the output, we can see ‘chinese’ is actually the most popular cuisine. And, we can do more cleaning on “\_id”:null, which you can see in “Issues in the Map” chapter.

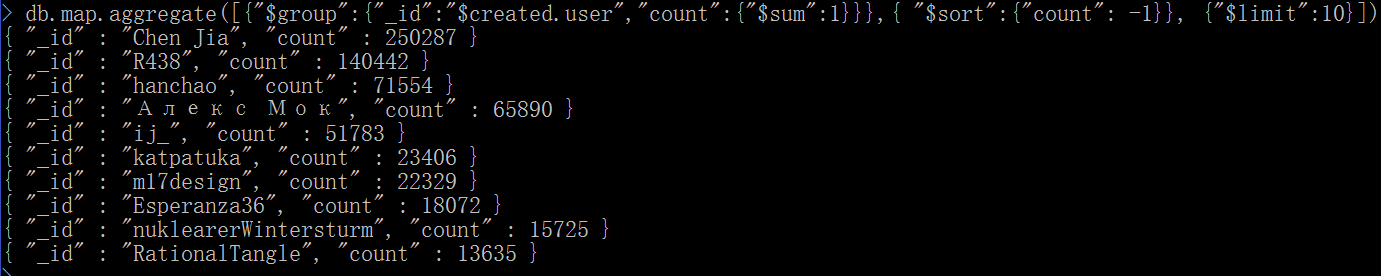
### Number of unique users

db.map.distinct("created.user").length



### Top N contributing user

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

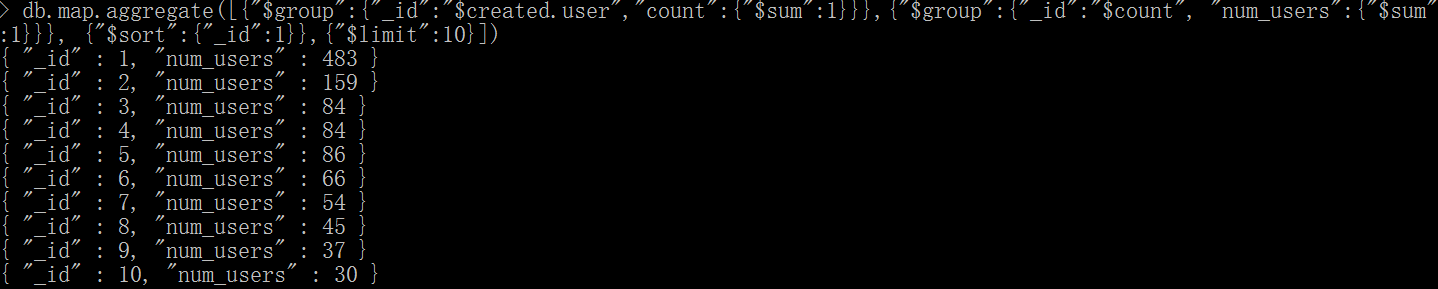


**I provide a rest service on** [**http://www.xingboedu:6000/topusers/<int:cnt**](http://www.xingboedu:6000/topusers/%3cint:cnt)**>**

### Number of users appearing only once (having <N post)

This idea comes from the sample project provided by Udacity.

db.map.aggregate([{"$group":{"\_id":"$created.user","count":{"$sum":1}}},{"$group":{"\_id":"$count", "num\_users":{"$sum":1}}}, {"$sort":{"\_id":1}},{"$limit":n}])



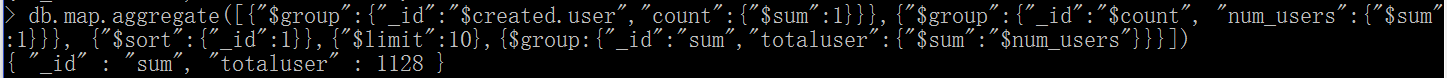
# “\_id” represents post count, here I list user numbers whose posts are less and equals 10.

### Contributor statistics

From the previous “Number of unique users“,”Top N contributing user” and “Number of users appearing only once” data, we can see the contribution of users are a little skewed.

1. **58% users contribute less or equal 10 posts**

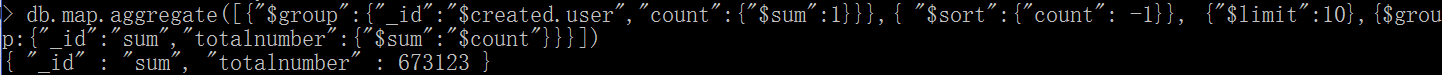
db.map.aggregate([{"$group":{"\_id":"$created.user","count":{"$sum":1}}},{"$group":{"\_id":"$count", "num\_users":{"$sum":1}}}, {"$sort":{"\_id":1}},{"$limit":n}, {$group:{"\_id":"sum","totaluser":{"$sum":"$num\_users"}}}])



The total number is 1128, 1128/1953≈58%.

1. **Top 10 users contribute percentage 64.5%**

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



The total number is 673123, 673123/1043322≈64.5%

I fully agree with the analysis in the sample project. If OpenStreetMap can make the map more lively(gamification) and provide some kinds of presents, such as scores, rewards, toys, It can make keep existing users and absorb more new users.

Also, a billboard can be provided in UI to stir users.

Next, I suppose if a user has mail attached, Corp can send survey by mail to know why they choose to use OpenStreetMap but not come frequently, and how they know OpenStreetMap, which may bring more users.

# Conclusion

After parsing, cleaning and analyzing the OSM file, it’s obvious that the map is incomplete, many locations are not included. But I am very happy to find there are more rows are added in recent years, it means more users know OpenStreetMap and begin to contribute it.

Also, issues in Beijing (Chinese string is used) is different from over-sea City (English is used) which is shown in our lesson. For example, abbreviation is often used in English street or road, whereas not in Chinese, for example:

“St” and “St.” –> “Street”

“街道” in Chinese has no abbreviation, it’s always “街道”

So it’s an issue in cities which use English, whereas not a big issue in cities which use Chinese.

Besides that, the map can be further cleaned, see what I mention in next Further chapter.

# Further

The project is a very simple project, I just write very simple script to parse and clean it. More energy can be used here to make the map more precise.

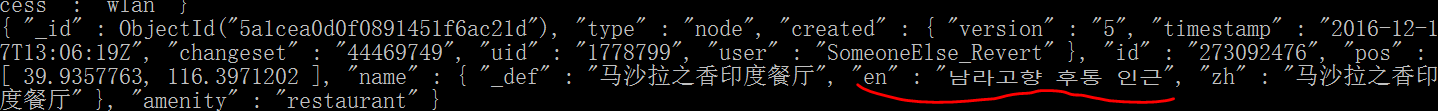
### Write more powerful fetchPostalCode method

In the project, I use the whole street value to search on the public web service, the result is not good. There are 11 rows with invalid postal code, just one row get postal code from the pub web service, the percentage is very low. The service supports fuzzy searching, so the best way is to just provide keywords to search. I need to let my script smarter to separate the whole Chinese string to keywords.

I know Elastic Search has Chinese analyzers，which I can make a reference, or even use ES in my project.

### Translation issue

During the analysis, I often browse the data randomly, somehow I suppose the setup of Roman kingdom cannot be without manual work. Then I found many random filled translation, for example: In this row, the name.en value is not English at all, it’s Korean string.



We may need further dig into it.

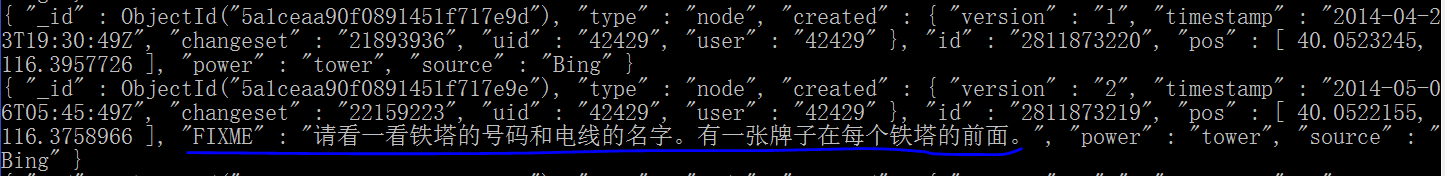
### Dispose relationship

Original OSM also contain <relationship> nodes, I have not dispose them. Somehow, it might include some interesting things to analysis

### Dispose FIXME Rows

When do analysis, I found some records’ FIXME field contain values which means current record has issue(s), this kind of records should be analyzed in further.

Next is such a record example:



### Build Knowledge Base

In the whole analysis process, I tried some public service providers; all of them are not fully free-charged: there is a daily calling upper limit.

So I have an idea to save the result in my own database, so in future, I can query in my own database before query public rest services.

Also, I want to save cleaned OpenStreetMap data in database, so it can be used to train robots.