OpenStreetMap Project
Data Wrangling with MongoDB

Tam, Francis

Map Area: Hong Kong (I live there)

https://www.openstreetmap.org/relation/913110#map=10/22.3527/114.1598
https://s3.amazonaws.com/metro-extracts.mapzen.com/hong-kong_china.osm.bz2

1. Problems Encountered in the Map

After downloading the data for Hong Kong area and do an initial auditing, we found that there are a couple of problems. E.g. Here are the unexpected street types found in audit.py:

```
{104 乡道: set([104 乡道]),
 106 侧: set([岭南路 106 侧]),
 168 号: set([广州市越秀区北站路 168 号]),
 20 号: set([沙坪镇园外苑 20 号]),
 20 號航天城高爾夫球場: set([赤鱲角香港國際機場航天城東路 20 號航天城高爾夫球場]),
 220 號: set([振安西路 220 號]),
 '260': set(['260']),
 2 路: set([佛平 2 路]),
 38-40 號: set([漢口道 38-40 號]),
 3 号: set([莲塘路 3 号]),
 '61-63': set(['61-63']),
 7 道: set([高新科技園高新南 7 道]),
'Aberdeen': set(['Old Main Street, Aberdeen']),
Amizade: set([友誼大馬路 Avenida da Amizade]),
Au: set([沙頭角公路 - 石涌凹段 Sha Tau Kok Road - Shek Chung Au]),
Av: set([坂雪岗大道 Bănxuě Găng Av]),
Ave: set([北环大道 North Ring Ave]),
'BLVD': set(['Shennan BLVD']),
Barbosa: set([巴波沙總督街 Rua Governador Tamagnini Barbosa]),
'Bay': set(['Castle Peak Road - Castle Peak Bay']),
'Bazaar': set(["Jardine's Bazaar"]),
Borja: set([青洲大馬路 (青洲新路) Avenida do Conselheiro Borja]),
Broadway: set(['Broadway', 百老匯街 Broadway]),
'Central': set(['Connaught Road Central',
                'Des Voeux Road Central',
                'Healthy Street Central',
                "Queen's Road Central",
                干諾道中 Connaught Road Central,
                皇后大道中 Queen's Road Central]),
'Chai': set(['230 Wan Chai Road, Wan Chai']),
Chau: set([鴨脷洲大街 Main Street, Ap Lei Chau]),
Chianti: set([尚堤 Chianti]),
Chung: set(['Mai Wo Industrial Building, 90-98 Kwai Cheong Road, Kwai Chung',
            青山公路 - 葵涌段 Castle Peak Road - Kwai Chung]),
Cinatti: set([爹美刁施拿地大馬路 (施拿地馬路) Avenida de Demétrio Cinatti]),
Circuit: set(['Chi Fuk Circuit',
              'Ho Hing Circuit',
              'Kin Fung Circuit',
              'Kwai Shing Circuit',
              'Tsing Hoi Circuit',
              'Tsing Yeung Circuit',
              'Tsuen King Circuit',
              'Yan Oi Tong Circuit',
              'Yuen Shun Circuit',
              源順圍 Yuen Shun Circuit]),
'Circuit\\': set(['Kwai Shing Circuit\\']),
'Close': set(['Kai Chi Close', 羅富國徑 Northcote Close]),
Coimbra: set([哥英布拉街 Rua de Coimbra]),
'Costa': set(['Avenida Horta e Costa']),
Crescent: set(['Greig Crescent',
               "Jardine's Crescent",
               'Wai Tsui Crescent',
               吉祥街 Kat Cheung Crescent,
               康樂園市中心徑 Hong Lok Yuen Town Centre Crescent,
               池旁路 Pond Crescent]),
Dadao: set([南山大道 Nanshan Dadao]),
'District': set(['Guanlan Longhua New District',
                 'Xili Town, Nanshan                  District']),
'East': set(['Canal Road East',
             'Expo Drive East',
             'Fu Mei Street East',
             'Fung Yau Street East',
             'Healthy Street East',
             'Kiu Cheong Road East',
             'Ping Ting Road East',
             'Prince Edward Road East',
             "Queen's Road East",
             'Sharp Street East',
             'Tai Wong Street East',
             'Wang Yip Street East',
             康樂東路 Hong Lok Road East,
             珠江东路 Zhujiang Road East,
             航天城東路 Sky City Road East,
             香山东街 Xiangshan Rd East]),
'Estate': set(['Cheung Shan Estate', 'Shek Wai Kok Estate']),
Felicidade: set([永福街 Rua da Felicidade]),
'Fong': set(['Siu Cheung Fong']),
Gaio: set([東望洋斜巷 Calçada do Gaio]),
'Glenealy': set(['Glenealy']),
 Guimarães: set([基馬拉斯大馬路 Avenida de Guimarães]),
 'Hang': set(['Wong Chuk Hang']),
 'Highway': set(["Hiram's Highway", 'Tsing Sha Highway']),
 'House': set(['Mody House']),
 'Hui': set(['Castle Peak Road - San Hui']),
 'Huizhou': set(['Tymphany Industrial Area Xin Lian Village, XinXu Town HuiYang District,
Huizhou']),
 Industrial: set([工業園前地 Praceta do Parque Industrial,
                 工業園北街 Rua Norte do Parque Industrial,
                 工業園大馬路 Avenida do Parque Industrial,
                 工業園新街 Rua Nova do Parque Industrial]),
 'It': set(['It']),
 'Kau': set(['Castle Peak Road - Ting Kau',
            'Tai Po Road - Tai Po Kau',
            大埔公路 - 大埔滘段 Tai Po Road - Tai Po Kau]),
 Keng: set([南京街 Rua de Nam Keng]),
 Kok: set([排角路 Rua do Pai Kok]),
 'Kowloon': set(['Salisbury Road, Kowloon']),
 Lacerda: set([罅些喇提督大馬路 (提督馬路) Avenida do Almirante Lacerda]),
 'Lam': set(['Castle Peak Road - Tai Lam']),
 Lin: set([大連街 Rua de Tai Lin]),
 'Load': set(['Guangfo Load']),
 'Long': set(['Castle Peak Road - Yuen Long',
             興隆街 Rua Heng Long]),
 'Lu': set(['Hua Fa Bei Lu',
           'Shenyan Lu',
           沙河西路 ShaHe Xi Lu,
           滨江东路 Binjiang Dong Lu]),
 'Mall': set(['Mount Sterling Mall']),
 Mau: set([林茂巷 Travessa de Lam Mau]),
 Mesquita: set([美上校圍 Pátio do Coronel Mesquita,
               美副將大馬路 Avenida do Coronel Mesquita]),
 Mok: set([和睦街 Rua Wo Mok]),
 Negra: set([黑橋街 Rua da Ponte Negra]),
 'North': set(['Chatham Road North',
              'Eastern Street North',
              'Fung Yau Street North',
              'Po Lam Road North',
              'Sai Cheung Street North',
              'Town Park Road North']),
 Oceano: set([海洋花園第六街 Rua Seis dos Jardins do Oceano]),
 Olímpica: set([奧林匹克大馬路 Avenida Olímpica]),
 On: set([信安馬路 Avenida Son On,
         北安前地 Largo de Pac On]),
 'One': set(['Siena Drive One']),
 'Park': set(['Hong Kong Park']),
 Patane: set([沙梨頭海邊街 Rua da Ribeira do Patane]),
 'Path': set(['Battery Path',
             'Chung Sing Path',
             'Glee Path',
             'Hau Yuen Path',
             'Hiu Yuk Path',
             'Holy Cross Path',
             'Kai Fat Path',
             'Kai Man Path',
             'Lee Fat Path',
             'Li Po Lung Path',
             'Ma Shing Path',
             'Mount Davis Path',
             'Ping Fai Path',
             'Rehab Path',
             'Shek Pai Tau Path',
             'Stadium Path',
             'Tramway Path',
             'Tsing Chui Path',
             'Tsing Ling Path',
             'Tsing Min Path',
             'Tsing Pak Path',
              'Tsing To Path',
              'Tsuen Hing Path',
              'Wah Lok Path',
              'Wai Yin Path',
              'Wo Hong Path',
              'Wong Chuk Hang Path',
              'Ying Choi Path',
              海浪徑 Hoi Long Path]),
Patos: set([鴨涌巷 Travessa do Canal dos Patos]),
Pereira: set([俾利喇街 Rua de Francisco Xavier Pereira]),
Pinto: set([飛能便度街 Rua Fernão Mendes Pinto]),
'Plaza': set(['Block A, DB Plaza']),
Prosperidade: set([順榮街 Rua da Prosperidade]),
'Queensway': set(['Queensway', 金鐘道 Queensway]),
'ROAD': set(['SO KWUN WAT ROAD']),
'Ranch': set(['Sea Ranch']),
Rd: set(['Cheong Wan Rd',
         'Gong ye 7th Rd',
         'Gongye 7th Rd',
         'Guihua Rd',
         'Hai yue Rd',
         'Houhaibin Rd',
         'Lize Rd',
         'Taikoo Shing Rd',
         中兴路 Zhongxing Rd,
         中山园路 Zhongshanyuan Rd,
         励耘路 Liyun Rd,
         延芳路 Yanfang Rd,
         文锦中路 Wenjin Middle Rd,
         新秀路 Xinxiu Rd,
         景山路 - Jǐngshān Rd,
         比亚迪路 Biyadi Rd,
         湖贝路 Hubei Rd,
         经二路 Jinger Rd,
         罗芳路 Luofang Rd,
         金风路 Jīnfēng Rd]),
Regedor: set([地堡街 Rua do Regedor]),
República: set([民國大馬路 Avenida da República]),
'Reservoir': set(['Family Walk, Ho Pui Reservoir']),
Riqueza: set([永富街 Rua da Riqueza]),
'Rise': set(['Consort Rise', "King's Park Rise"]),
'Row': set(['Minden Row']),
'S.': set(['Gaoxin S.']),
Seca: set([船澳街 Rua da Doca Seca]),
Sen: set([孫逸仙博士大馬路 Avenida Dr. Sun Yat Sen]),
'Shan': set(['Castle Peak Road - Ping Shan']),
'Shui': set(['Tai Po Road - Ma Liu Shui']),
Silva: set([伯多祿局長街(白馬行) Rua de Pedro Nolasco da Silva]),
'Smithfield': set(['Smithfield']),
'South': set(['Chatham Road South',
              'Container Port Road South',
              'Fung Yau Street South',
              'Gillies Avenue South',
              'Po Lam Road South',
              'Sai Yeung Choi Street South',
              'Town Park Road South',
              'Wang Yip Street South',
              西洋菜南街 Sai Yeung Choi Street South]),
'St': set(["d'Aguilar St", 仙湖路 Xianhu St]),
'St.': set(['Finance St.']),
'Station': set(['East Tsim Sha Tsui Station']),
Strand: set(['Bonham Strand', 文咸東街 Bonham Strand]),
Street: set([Wang Lung Street]),
Sul: set([海灣南街 Rua da Bacia Sul]),
'Tau': set(['Castle Peak Road - Tsing Lung Tau']),
'Terrace': set(['Dragon Terrace',
                'Kai Yuen Terrace',
                'Moreton Terrace',
                'Sai Wan Terrace',
                'Ying Wa Terrace']),
'Tin': set(['Castle Peak Road - San Tin']),
'Tong': set(['Ko Tong', 'Luk Tei Tong']),
'Town': set(['New Praya, Kennedy Town', 'Praya, Kennedy Town']),
Tranquilidade: set([永寧街 Rua da Tranquilidade]),
'Tsai': set(['O Tsai', 'Tai Po Road - Yuen Chau Tsai']),
'Tseng': set(['Castle Peak Road - Sham Tseng']),
'Tsuen': set(['Wong Chuk Shan San Tsuen']),
'Twisk': set(['Route Twisk']),
'Two': set(['Siena Drive Two']),
Verde: set([青洲上街 Rua da Ilha Verde,
            青洲新馬路 Estrada Nova da Ilha Verde,
            青洲河邊馬路 Estrada Marginal da Ilha Verde]),
'Village': set(['Parkridge Village']),
Vitória: set([得勝馬路 Estrada da Vitória]),
'Wan': set(['Castle Peak Road - Tsuen Wan']),
West: set(['Austin Road West',
           'Catering Road West',
           'Connaught Road West',
           'Des Voeux Road West',
           'Healthy Street West',
           'Hing Wah Street West',
           'Prince Edward Road West',
           "Queen's Road West",
           'Wang Yip Street West',
           'Yen Chow Street West',
           太子道西 Prince Edward Road West,
           干諾道西 Connaught Road West,
           康樂西路 Hong Lok Road West,
           德輔道西 Des Voeux Road West,
           柯士甸道西 Austin Road West,
           珠江西路 Zhujiang Road West,
           皇后大道西 Queen's Road West]),
'Wo': set(['Tai Po Road - Tai Wo']),
'Xi': set(['Huang Pu Dadao Xi']),
'Yuen': set(['Po Wah Yuen']),
'kamtin': set(['kamtin']),
'road': set(['Huangguan road',
             民田路 Mintian road,
             福华路 Fuhua road]),
'zi': set(['Tai zi'])}
```

Some of them are actually legit, which reflects the cultural difference or some specific localized rules in naming the streets. Some of them can be improved. All of them can be further classified into several sub-categories:

- Abbreviated street types (“Cheong Wan Rd”)
- Unusual street types (“Holy Cross Path”)
- Street names in Unicode

Data with proper street type translations are translated when transforming the XML data to json.

Abbreviated Street Types

Some street types are abbreviated. E.g. Rd can be changed to Road, Av can be changed to Avenue.

Unusual Street Types

E.g. “Path” can be used in Hong Kong, e.g. “Holy Cross Path”. Roads ending with a direction word is acceptable and often used, e.g. “Des Voeux Road West”. If the street type is legit, they are not cleaned up before importing to MongoDB.

Street Names in Unicode

E.g. “u'104\u4e61\u9053'” before we refined the pprint output (which is “104 乡道” in utf-8). If they are translated to English properly, they can be one of the legit street types. However, some of them have English translation attached and they are obviously some street names from Macao (in Portuguese, e.g. “Rua de Francisco Xavier Pereira'”) or mainland China (they translate street name to English using one word, e.g. they will use “Xinxiu Rd'” in mainland China but we will use “Xin Xiu Rd'” in Hong Kong.)

City

Since some addresses are suspected to be in another city, we conducted more queries after importing the data to MongoDB (codes in subsequent section):

```
Pipeline = [
     {"$match":{"address.city":{"$exists":1}}},
     {"$group":{"_id":"$address.city", "count":{"$sum":1}}},
     {"$sort":{"count":-1}},
     {"$limit":10}
]
```

```
{u'count': 363, u'_id': 香港 Hong Kong}
{u'count': 53, u'_id': Sai Kung}
{u'count': 39, u'_id': Zhuhai}
{u'count': 27, u'_id': 广州}
{u'count': 25, u'_id': 深圳 Shenzhen}
{u'count': 24, u'_id': Hong Kong}
{u'count': 22, u'_id': 深圳市}
{u'count': 15, u'_id': Shenzhen}
{u'count': 13, u'_id': guangzhou}
{u'count': 11, u'_id': 广州市}
```

```
    {u'count': 10, u'_id': 澳門 Taipa}
```

So we found that 39 documents are in fact means the city “Zhuhai”, 25 + 15 in “Shenzhen” and 13 in “guangzhou” which are obviously cities in mainland China. The wrong ones will be filtered when exporting to the .json file

Phone

The wrong city issue can be further confirmed from getting the phone numbers:

```
    Pipeline = [
        {"$match":{"phone":{"$exists":1}}},
        {"$group":{"_id":"$phone", "count":{"$sum":1}}},
        {"$sort":{"_id":-1}},
        {"$limit":100}
    ]
```

Some results:

```
    {u'count': 1, u'_id': u'+86 020 8337 3868'}
    {u'count': 1, u'_id': u'+86 020 81048888'}
    {u'count': 1, u'_id': u'+86 020 38145775'}
    {u'count': 1, u'_id': u'+85328833762'}
    {u'count': 1, u'_id': u'+853 8898 9617'}
    {u'count': 1, u'_id': u'+853 8898 9593'}
    {u'count': 1, u'_id': u'+853 8898 1627'}
    {u'count': 1, u'_id': u'+853 8898 1501'}
```

+86 is from mainland China, +853 is Macao. (Hong Kong is +852).

All the above are cleaned up before importing to MongoDB. Please refer to data.py, def is_hongkong() for the detail filtering procedures.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

```
hong-kong_china.osm ......... 421.268 MB
hong-kong_china.osm.json .... 648.977 MB
```

# Number of tags in the initial .osm file – no unusual tag spotted

Results:
```
{'bounds': 1,
 'member': 61904,
 'nd': 2381157,
 'node': 2076881,
 'osm': 1,
 'relation': 5674,
 'tag': 689560,
 'way': 202653}
```

Code:
```
import xml.etree.ElementTree as ET
import pprint

def count_tags(filename):
        tags={}
        for event, elem in ET.iterparse(filename):
            if elem.tag in tags:
                tags[elem.tag]+=1
            else:
                tags[elem.tag]=1

        return tags

def test():
    tags = count_tags('hong-kong_china.osm')
    pprint.pprint(tags)

if __name__ == "__main__":
    test()
```

# Analysis of attribute ‘k’ in the tag “tag” in .osm – no problematic characters

Results:
```
{'lower': 544458, 'lower_colon': 135420, 'other': 9682, 'problemchars':
0}
```

Code:
```
import xml.etree.ElementTree as ET
import pprint
import re

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

def key_type(element, keys):
    if element.tag == "tag":
        if lower.match(element.attrib['k']):
            keys['lower']+=1
        elif lower_colon.match(element.attrib['k']):
            keys['lower_colon']+=1
        elif problemchars.match(element.attrib['k']):
            keys['problemchars']+=1
        else:
            keys['other']+=1
    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)
    return keys

def test():
    keys = process_map('hong-kong_china.osm')
    pprint.pprint(keys)

if __name__ == "__main__":
    test()
```

If `ET.tostring(element)` is added to see what is in "others", here are some examples:

```
<tag k="seamark:harbour:category" v="marina" />
<tag k="destination:zh-yue" v="&#39340;&#22580;" />
<tag k="gns:ADM1" v="30" />
```

So they have either more than one colon, a hyphen, or capital letters. They still seem to be legit formats so we simply proceed.

# Analysis of users contributed in .osm – Unicode characters spotted

Sample Results:

```
      u'\u4e5d\u5df4\u53ca\u57ce\u5df4',
      u'\u4eba\u6c11\u597d\u5144\u5f1f',
      u'\u521b\u673a\u4f1a',
```

Obviously they are in Unicode and we might need to bear it in mind when analyzing or even processing the data. E.g. It needs to be corrected by customizing pprint.

Code:

```
import xml.etree.ElementTree as ET
import pprint
import re

def get_user(element):
    return

def process_map(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if element.tag=="node":
            users.add(element.attrib['user'])
    return users

def test():
    users = process_map('hong-kong_china.osm')
    pprint.pprint(users)

if __name__ == "__main__":
    test()
```

# Code used to make queries on MongoDB after importing the .json file (Database: OpenStreetMap, Collection: Hong Kong)

```
def get_db(db_name):
    from pymongo import MongoClient
    client = MongoClient('localhost:27017')
    db = client[db_name]
    return db

def make_pipeline():
    # the aggregation pipeline if needed
    pipeline = []
    return pipeline

def aggregate(db, pipeline):
    result = db.HongKong.aggregate(pipeline)
    return result

if __name__ == '__main__':
    db = get_db('OpenStreetMap')

    # Queries area:
    # e.g. print db.HongKong.find().count()
    # if aggregation needed:
    # pipeline = make_pipeline()
    # result = aggregate(db, pipeline)
    # for doc in result:
    #     print doc
```

# Number of documents

```
print db.HongKong.find().count()
2279115
```

# Number of nodes

```
print db.HongKong.find({"type":"node"}).count()
2074978
```

# Number of ways

```
print db.HongKong.find({"type":"way"}).count()
201975
```

# Number of unique users

```
print len(db.HongKong.distinct("created.user"))
1187
```

# Top 10 contributing users

```
Pipeline = [
    {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
    {"$sort":{"count":-1}},
    {"$limit":10}
]

{u'count': 517121, u'_id': u'hlaw'}
{u'count': 212849, u'_id': u'MarsmanRom'}
{u'count': 165087, u'_id': u'Popolon'}
{u'count': 101103, u'_id': u'fsxy'}
{u'count': 91243, u'_id': u'katpatuka'}
{u'count': 74309, u'_id': u'KX675'}
```

```
{u'count': 71466, u'_id': u'fdulezi'}
{u'count': 68855, u'_id': u'sn0wblind'}
{u'count': 56517, u'_id': u'rainy3519446'}
{u'count': 45709, u'_id': u'bTonyB'}
```

In total they contributed 1404259 documents, roughly 2/3 of all.

# Number of users appearing only once (having 1 post)

```
Pipeline = [
     {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
     {"$group":{"_id":"$count", "num_user":{"$sum":1}}},
     {"$sort":{"_id":1}},
     {"$limit":1}
]

{u'_id': 1, u'num_user': 211}
# "_id" represents postcount
```

3. Additional Ideas

Deploying Bots

When doing more queries (like those in below), we will find that many nodes are with only poor information. E.g. the 2nd top religion in Hong Kong is "None", the top cuisine in Hong Kong is "None", and the top restaurant in Hong Kong is again "None".

It means that the data of Hong Kong in Open Street Map is still incomplete.

Why?

While Open Street Map is open source, demand drives supply. If the utilization of it in the city grows, there will be more contributors naturally. In particular to Hong Kong, people use a lot of mobile apps and a lot of them built on locations. If we can entice those apps to utilize Open Street Map and ask them feedback with more accurate information, the data quality will improve.

But the challenge is that if good data does not even exist, nobody will ever use it and contribute, then the data will remain poor forever. This is a vicious cycle. Then how can we solve this problem?

We might need to deploy bot to crawl data from others.

Usually local government / agencies will provide such free data. In particular to Hong Kong, there is source from the Hong Kong Government: http://www2.map.gov.hk/gih3/view/index.jsp. In the Map Tools there, we can download .csv, transform the data, then import back to the OSM database.

E.g. the format for all libraries in HK:

"ENGLISH CATEGORY"	"中文類別"	"ENGLISH NAME"	"中文名稱"	"ENGLISH ADDRESS"	"中文地址"	"LONGITUDE""經度"	"LATITUDE"	"緯度"	"EASTING"	"坐標東"	"NORTHING"	"坐標北"	"DISTRICT"	"地區"	"LIBRARY TYPE"	"圖書館種類"	"OPENING HOURS"	"開放時間"	"TELEPHONE"	"聯絡電話"	"FAX NUMBER"	"傳真號碼"	"EMAIL ADDRESS"	"電郵地址"	"WEBSITE"	"網頁"	"MLNO"	"MLNO"
"Libraries"	"圖書館"	"Tai Wo Estate(Mobile Library 5)"	"太和邨(流動圖書館五)"	"Adjacent to Tai Wo Shopping Arcade, Oi Wo House, Tai Wo Estate, Tai Po, N.T."	"新界大埔太和邨愛和樓太和商場出口"	"114-9-42"	"114-9-42"	"22-27-8"	"22-27-8"	"834694.1"	"834694.1"	"834758.9"	"834758.9"	"TAI PO"	"大埔"	"MOBILE LIBRARY"	"流動圖書館"	""	""	""	""	""	""	""	""	"http://www.hkpl.gov.hk/en/locations/tai-po/mobile5/adjacent-to-tai-wo.html"	"http://www.hkpl.gov.hk/tc/locations/tai-po/mobile5/adjacent-to-tai-wo.html"	"ML05""ML05"

There are much more amenities data there. We can use them to validate or add the existing data in Open Street Map.

Of course it is up to Open Street Map to explore the locations and implement such bots to beef up their data, which needs resources. So the challenge is not really whether we have the data, but how we motivate people, or whether we can put in resources, to get the data.

No matter how good an analysis is, the end result is as good as the quality of the original data. Improving it is the first step before everything. This is too early to say the data can be used in other projects.

Additional data exploration using MongoDB queries

# Top 20 appearing amenities

```
Pipeline = [
      {"$match":{"amenity":{"$exists":1}}},
      {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
      {"$sort":{"count":-1}},
      {"$limit":20}
]

{u'count': 1547, u'_id': u'parking'}
{u'count': 1307, u'_id': u'school'}
```

```
{u'count': 992, u'_id': u'toilets'}
{u'count': 629, u'_id': u'bus_station'}
{u'count': 583, u'_id': u'restaurant'}
{u'count': 574, u'_id': u'shelter'}
{u'count': 395, u'_id': u'bank'}
{u'count': 394, u'_id': u'place_of_worship'}
{u'count': 271, u'_id': u'fuel'}
{u'count': 246, u'_id': u'fast_food'}
{u'count': 219, u'_id': u'taxi'}
{u'count': 198, u'_id': u'bicycle_parking'}
{u'count': 188, u'_id': u'hospital'}
{u'count': 174, u'_id': u'marketplace'}
{u'count': 169, u'_id': u'swimming_pool'}
{u'count': 157, u'_id': u'cafe'}
{u'count': 117, u'_id': u'public_building'}
{u'count': 107, u'_id': u'police'}
{u'count': 107, u'_id': u'library'}
{u'count': 106, u'_id': u'community_centre'}
```

# Top religions

```
Pipeline = [
     {"$match":{"amenity":{"$exists":1},"amenity":"place_of_worsh
ip"}},
     {"$group":{"_id":"$religion", "count":{"$sum":1}}},
     {"$sort":{"count":-1}},
     {"$limit":10}
]

{u'count': 137, u'_id': u'christian'}
{u'count': 109, u'_id': None}
{u'count': 75, u'_id': u'buddhist'}
{u'count': 66, u'_id': u'taoist'}
{u'count': 4, u'_id': u'muslim'}
{u'count': 1, u'_id': u'sikh'}
{u'count': 1, u'_id': u'multifaith'}
{u'count': 1, u'_id': u'*'}
```

Interestingly, we see “None” here, probably due to the fact that they cannot be classified properly, with lots of stream of religions in diverse culture in China.

# Most popular cuisines

```
Pipeline = [
     {"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}},
     {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
		{"$sort":{"count":-1}},
		{"$limit":10}
	]

	{u'count': 339, u'_id': None}
	{u'count': 111, u'_id': u'chinese'}
	{u'count': 15, u'_id': u'indian'}
	{u'count': 15, u'_id': u'japanese'}
	{u'count': 12, u'_id': u'regional'}
	{u'count': 8, u'_id': u'italian'}
	{u'count': 7, u'_id': u'pizza'}
	{u'count': 6, u'_id': u'cantonese'}
	{u'count': 5, u'_id': u'french'}
	{u'count': 5, u'_id': u'international'}
```

# Top restaurants – restaurant, café, or fast food

```
	Pipeline = [
		{"$match":{"amenity":{"$exists":1},"$or":[{"amenity":"restau
	rant"}, {"amenity":"fast_food"}, {"amenity":"cafe"}]}},
		{"$group":{"_id":"$name", "count":{"$sum":1}}},
		{"$sort":{"count":-1}},
		{"$limit":10}
	]

	{u'count': 118, u'_id': None}
	{u'count': 80, u'_id': u"McDonald's"}
	{u'count': 38, u'_id': u'KFC'}
	{u'count': 20, u'_id': u'Starbucks'}
	{u'count': 15, u'_id': u'Pizza Hut'}
	{u'count': 9, u'_id': u'McDonalds'}
	{u'count': 8, u'_id': u'Starbucks Coffee'}
	{u'count': 7, u'_id': u'Pacific Coffee'}
	{u'count': 7, u'_id': u'Burger King'}
	{u'count': 6, u'_id': u'Subway'}
```

While the top restaurants are all fast-food shops like Starbuck, KFC, and McDonald which is not surprising (Hong Kong people really loves fast food), the number of them shows problem of the data. In Hong Kong, there are easily more than 200 McDonalds (http://www.answers.com/Q/How_many_McDonald%27s_restaurants_are_in_Hong_Kong) but many of them are in shopping malls – they are not in standalone buildings. OpenStreetMap can only show data in nodes, but not in deeper details. So we need to be very careful in doing similar analysis in cities like Hong Kong.

Conclusion

After this review of the data it's obvious that the Hong Kong area is incomplete and inaccurate. Top problems are:

1. It includes nodes in nearby cities like Shenzhen in mainland China.
2. Most of the nodes do not have good enough information like what cuisine the restaurant is serving or what religion the building is devoted to.
3. In Hong Kong, many buildings are composite buildings like shopping mall at the same time there are residential building on top. We cannot represent them correctly in Open Street Map and we cannot tell what are inside there. Perhaps we need a better second level representation in the amenities (e.g. in a shopping mall, there is KFC, McDonald, etc.) so that it is more useful for users/apps to take reference to, in turn driving up demand.

While the first issue is cleaned up when importing to MongoDB, the second and third issues are no easy fix. The second one is related to fundamental data quality and bots was suggested to help solving it, but the third one is related to the design of OpenStreetMap which needs more mindful consideration. E.g. if we add second level representation, does it fit the other cities as well? Will the same "key" name be understood as something else in other countries? They need to be addressed before any next release.