OpenStreetMap Sample Project
Data Wrangling with MongoDB

Feby Thomas

Map Area: Milwaukee, WI, United States

https://s3.amazonaws.com/metro-extracts.mapzen.com/milwaukee_wisconsin.osm.bz2

Contents

1. Problems Encountered in the Map

Downloaded the Milwaukee, WI, US area OSM xml and found the following key problems:

- Inconsistent street names like Ave, Ave., Ct, D, Dr etc
- There were incorrect postcode values for some like: (“Milwaukee WI, 53222”, “1729”, “WI”)
- Inconsistent state entries: 'Milwaukee', 'WII', 'Wisconsin'
- Based on City names, it is clear that it belongs to the Milwaukee–Racine–Waukesha metropolitan area

Inconsistent Street & State Names

Before importing the data to Mongo DB I cleaned up the Inconsistent street names and the state entries, the logic for the same can be found in mapclean.py
There are couple of street names which has fully wrong values like “53076” which might need manual fixing based on exact knowledge using command like:

```
> db.osmdatamke.update({"address.postcode":"1729"},{"$set": {"address.postcode": "53076", "address.city":"Richfield", "address.street":"1729 Wolf Road"}})
```

Additionally, there were multiple addressed without the state tag for address, which in this case updated to WI using the below query:

```
> db.osmdatamke.update({"address.street":{"$exists":1},"address.state":{"$exists":false}},{ "$set":{"address.state":"WI"}}, {multi: true})
```

```
> db.osmdatamke.update({"address.street":{"$exists":1},"address.state":{"$exists":false}},{ "$set":{"address.state":"WI"}}, {multi: true})
WriteResult({ "nMatched" : 1017, "nUpserted" : 0, "nModified" : 1017 })
```

Postal Codes

The postal codes which had issue were only 1 per type, so that was updated after importing in the Mongo DB:

```
{ "_id" : "1729", "count" : 1 }
{ "_id" : "WI", "count" : 1 }
{ "_id" : "Milwaukee WI, 53222", "count" : 1 }
```

```
> db.osmdatamke.update({"address.postcode":"Milwaukee WI, 53222"},{"$set": {"address.postcode": "53222" }})
```

Before:

```
> db.osmdatamke.find({"address.postcode":"1729"}).pretty()
{
        "_id" : ObjectId("556986550539ef2a2ce285fe"),
        "node_refs" : [
                "1768286824",
                "1768286826",
                "1768286802",
                "1768286842",
                "1768286836",
                "1768286804",
                "1768286849",
                "1768286823",
                "1768286834",
                "1768286840",
                "1768286841",
                "1768286824"
        ],
        "pos" : [ ],
        "cuisine" : "american",
        "type" : "way",
        "amenity" : "restaurant",
        "id" : "165272758",
        "building" : "yes",
        "created" : {
                "timestamp" : "2012-05-28T05:02:54Z",
                "changeset" : "11722497",
                "version" : "2",
                "uid" : "5611",
                "user" : "alexrudd"
        },
        "address" : {
                "street" : "53076",
                "postcode" : "1729"
        },
        "name" : "Sawmill Inn"
}
```

After update:

```
> db.osmdatamke.find({"alt_name" : "Saint Aemillian Home"}).pretty()
{
        "_id" : ObjectId("556985900539ef2a2cdce291"),
        "pos" : [
                43.0889003,
                -88.0239772
        ],
        "ele" : "238",
        "alt_name" : "Saint Aemillian Home",
        "type" : "node",
        "id" : "367808386",
        "building" : "yes",
        "created" : {
                "uid" : "1533276",
                "changeset" : "16894017",
                "version" : "3",
                "timestamp" : "2013-07-10T00:26:56Z",
                "user" : "cncr04s"
        },
        "address" : {
                "state" : "WI",
                "street" : "West Capitol Drive",
                "postcode" : "53222",
                "housenumber" : "8901"
        },
        "source" : "USGS Geonames"
}
```

The 9-Digit postal codes were left as-is.

Sort postcodes by count, descending

The top postal codes seem to be fine and part of the greater Milwaukee area.

```
> db.osmdatamke.aggregate([{"$match": {"address.postcode":{"$exists":1}}},{"$group":
{"_id":"$address.postcode","count":{"$sum":1}}},{"$sort":{"count":-1}}])
```

2. Data Overview

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

File sizes

milwaukee_wisconsin ..... 129 MB
milwaukee_wisconsin.osm.json .... 182 MB

Number of documents

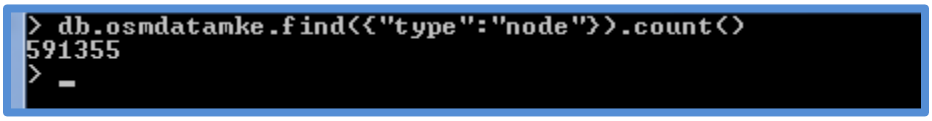
```
print(db.osmdatamke.find().count())
```

655018

Number of nodes

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

591355



Number of ways

print(db.osmdatamke.find({"type":"way"}).count())
63649

```
> db.osmdatamke.find({"type":"way"}).count()
63649
```

Number of unique users

```
> db.osmdatamke.distinct("created.user").length
582
```

Top 1 contributing user

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

```
> db.osmdatamke.aggregate([
...                                     {"$group":{"_id":"$created.user",
...                                                "count":{"$sum":1}}},
...                                     {"$sort":{"count":-1}},
...                                     {"$limit" : 1}])
{ "_id" : "woodpeck_fixbot", "count" : 184700 }
```

Number of users appearing only once (having 1 post)

>db.osmdatamke.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}},{"$group": {"_id": "$count", "num_users": {"$sum":1}}},{"$sort":{"_id":1}},{"$limit" : 1}])

```
> db.osmdatamke.aggregate([
...                                     {"$group":{"_id":"$created.user",
...                                                "count":{"$sum":1}}},
...                                     {"$group": {"_id": "$count", "num_users": {"$sum":1}}},
...                                     {"$sort":{"_id":1}},
...                                     {"$limit" : 1}])
{ "_id" : 1, "num_users" : 122 }
```

Number of schools in the area: 851

> db.osmdatamke.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"school"}}, {"$group":{"_id":"$amenity", "count":{"$sum":1}}},{"$sort":{"count":-1}}])

```
> db.osmdatamke.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"school"}}, {"$group":{"_id
":"$amenity", "count":{"$sum":1}}},        {"$sort":{"count":-1}}])
{ "_id" : "school", "count" : 851 }
```

Top shops in the area:

> db.osmdatamke.aggregate([{"$group":{"_id":"$shop", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])

```
> db.osmdatamke.aggregate([{"$group":{"_id":"$shop", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])
{ "_id" : null, "count" : 654529 }
{ "_id" : "supermarket", "count" : 91 }
{ "_id" : "clothes", "count" : 46 }
{ "_id" : "convenience", "count" : 46 }
{ "_id" : "car_repair", "count" : 29 }
{ "_id" : "doityourself", "count" : 27 }
{ "_id" : "department_store", "count" : 18 }
{ "_id" : "car", "count" : 17 }
{ "_id" : "hairdresser", "count" : 17 }
{ "_id" : "alcohol", "count" : 12 }
{ "_id" : "mall", "count" : 10 }
{ "_id" : "jewelry", "count" : 8 }
{ "_id" : "bicycle", "count" : 8 }
{ "_id" : "shoes", "count" : 7 }
{ "_id" : "mobile_phone", "count" : 7 }
{ "_id" : "laundry", "count" : 6 }
```

3. Additional Ideas

The majority of entries are latitude and longitudes, with address related nodes being only 1283:

```
> db.osmdatamke.distinct("address").length
1283
```

And most of the node related entries are done by user: “woodpeck_fixbot” which looks to be some automated mechanism of update.

```
> db.osmdatamke.aggregate([                                    {"$group":{"_id":"$created.user",
                                                "count":{"$sum":1}}},
 {"$sort":{"count":-1}}])
{ "_id" : "woodpeck_fixbot", "count" : 184700 }
{ "_id" : "Gary Cox", "count" : 35055 }
{ "_id" : "ItalianMustache", "count" : 34640 }
{ "_id" : "bbauter", "count" : 33877 }
{ "_id" : "iandees", "count" : 28811 }
{ "_id" : "hogrod", "count" : 26360 }
{ "_id" : "TIGERcnl", "count" : 22143 }
```

Additional data exploration using MongoDB queries

Top 10 appearing amenities

> db.osmdatamke.aggregate([{"$group":{"_id":"$amenity", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])

```
> db.osmdatamke.aggregate([{"$group":{"_id":"$amenity", "count":{"$sum":1}}}, {"$sort":{"count":-1}}
])
{ "_id" : null, "count" : 651057 }
{ "_id" : "parking", "count" : 1237 }
{ "_id" : "school", "count" : 851 }
{ "_id" : "restaurant", "count" : 288 }
{ "_id" : "fast_food", "count" : 177 }
{ "_id" : "grave_yard", "count" : 157 }
{ "_id" : "fuel", "count" : 144 }
{ "_id" : "place_of_worship", "count" : 111 }
{ "_id" : "cafe", "count" : 74 }
{ "_id" : "toilets", "count" : 65 }
{ "_id" : "bank", "count" : 65 }
{ "_id" : "emergency_phone", "count" : 63 }
{ "_id" : "bar", "count" : 63 }
{ "_id" : "hospital", "count" : 56 }
{ "_id" : "fire_station", "count" : 55 }
{ "_id" : "post_office", "count" : 52 }
{ "_id" : "library", "count" : 49 }
{ "_id" : "pub", "count" : 41 }
{ "_id" : "townhall", "count" : 40 }
{ "_id" : "shelter", "count" : 34 }
```

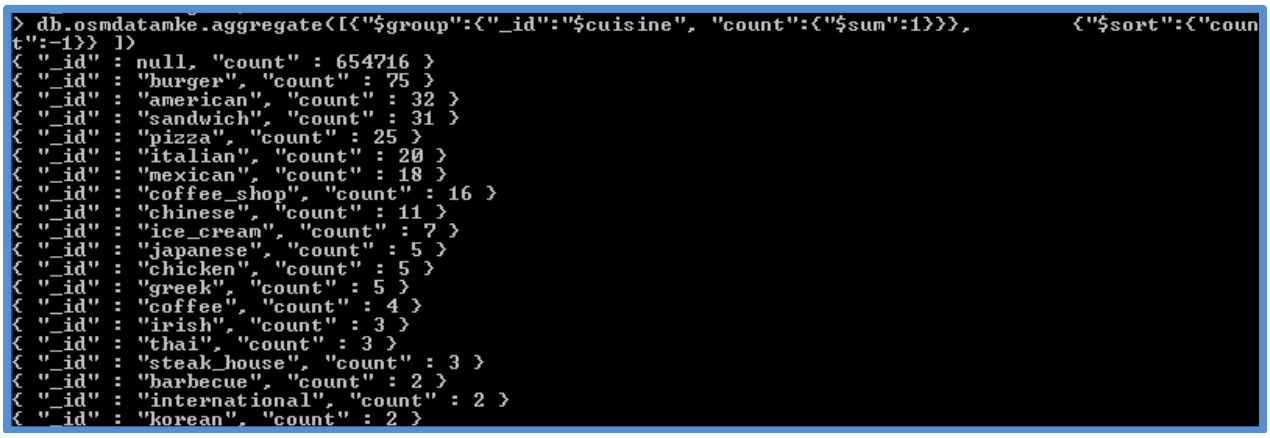
The other thing which is evident for future update of this map is that ATMs in the whole of greater Milwaukee area is only shown as 15:

```
{ "_id" : "atm", "count" : 15 }
```

This surely is not correct and indicates scope for further improvement in the OSM data that we can contribute.

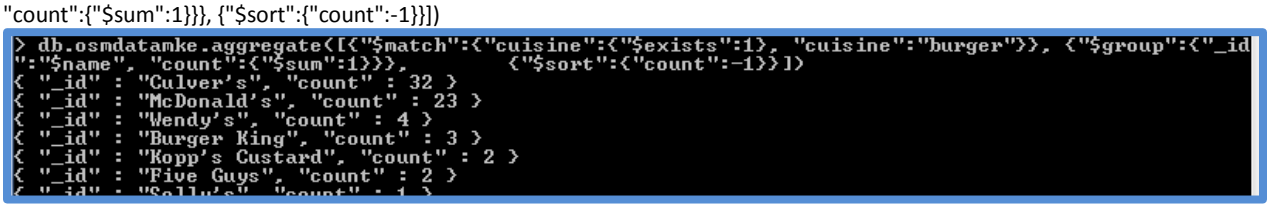
Most popular cuisines

> db.osmdatamke.aggregate([{"$match":{"cuisine":{"$exists":1}}}, {"$group":{"_id":"$name", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])



Fast food seems to be the most common cuisine and also surprising to note that **“Culver’s”** Burger joint has the largest presence in the Milwaukee area.

> db.osmdatamke.aggregate([{"$match":{"cuisine":{"$exists":1}, "cuisine":"burger"}}, {"$group":{"_id":"$name", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])



Top shops in the area

> db.osmdatamke.aggregate([{"$match":{"shop":{"$exists":1}}}, {"$group":{"_id":"$name", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])

Conclusion

After this review of the data it is my understanding that the region covered for greater Milwaukee area is quite accurate. There is definitely scope for much more improvement; more importantly utility related amenities like ATMS seems to be too less for such a large area and that could be easily updated with help of information from banks and their sites or with screen scraping. Same seems to be the case shops (Walmart, Target etc) and famous restaurants which look like grossly under updated, and this something

which is most searched for, this could be easily crowdsourced using GPS info and user mobiles, by incentivizing for updates.

Bank ATM Update:

Approach:

The current list of ATMs seems to be too less for such a large area, the recommended approach above is to use the bank sites and screen scraping to get the required details.

Anticipated Problems:

We would need to overcome the following challenges:

1. Each bank has different site, we will need slightly modified script for each bank.
2. We will need to build logic for each bank separately
3. Then we will also need to search for ATMs near unique pincodes in the Milwaukee Greater Area and then remove duplicates if any.
4. There is also a possibility to get some of the ATMs using the Visa and Master Card ATM locator APIs, but this may not provide full coverage of all ATMs:
    a. https://developer.visa.com/atmlocator
    b. https://developer.mastercard.com/portal/display/api/Locations

Shop Location Update:

Approach:

The suggestion for this was to crowdsource this using a easy to use approach of Mobile GPS location and simple info nearby. This approach will need the geo-spatial indexes and queries for streets around it for exact address. The approach similar to the bank site scraping can also be used here to but it will have similar challenges as above.

Anticipated Problems:

The approach could have the following challenges:

1. Engaging the crowdsourcing without incentive or funding could be a challenge
2. Finding the exact address like the plot number could be a challenge in this approach.
3. Finding additional attributes like the shop type (super market, specialty etc) or restaurant specialty / cuisine could be a challenge