OpenStreetMap Project
Data Wrangling with MongoDB

Saeideh Shahrokh Esfahani

Map Area: San Francisco, CA, United States

I have downloaded the .osm file for San Francisco area. The file was almost 1 GB. Therefore, in order to develop my code, I generated a sample of my .osm file using data_sample1.py. The generated sample file is "san-francisco_test1.osm".

1. Problems Encountered in the Map

In order to clean the data, I address the following issues:

1) **Over-abbreviated street names** (e.g. Sansome St #3500).
   I have used audit_street_name.py inorder to extract probable abbreviation.
   Then I wrote a function in order to overcome with this issue. The final result is:
   Sansome St #3500 => Sansome Street #3500
   Willow Rd => Willow Road
2) **Cleaning the "name" from different issues like additional name inside the parentheses or & sign.**
   After running audit_name.py, I noticed the problems and wrote the improve_name.py to check the cleaning function. The final result is:
   Piedmont Baths (historical) => Piedmont Baths
   Mio Spa & Salon => Mio Spa and Salon

3) **Replace "_" with " " in amenity field**.
   Using audit_amenity.py I checked my cleaning function related to this field.
   Fast_Food => Fast Food
   Ice_Cream => Ice Cream
4) **Removing CA after county name.**
   I created audit_county.py to remove CAs after the county name
   Alameda, CA => Alameda
   San Mateo, CA => San Mateo
5) **Creating a dictionary for the timestamp and transfer string date to date**.
   I used "dateutil.parser" in order to extract the year, month and date from the value of the timestamp.
   timestamp="2016-01-26T16:14:53Z" => timestampe = { year: 2016 , month: 01 , day: 26}

2. Preparing the JSON file

In this part, I locked up the model by which I would represent my JSON file. Using the code from the last section of “Case Study” and the function which I wrote for the cleaning of the data, I implemented create_json.py. Before working with the main file, again, I generated the JSON sample file under the name of san-francisco_test1.osm.json for analyzing my query. Then I created the main JSON file and called it New_san-francisco_california.osm.json.

3. Importing the data in MongoDB with Mongoimport

I used the following command in order to import my data into the database:

```
mongoimport --db final_os --collection finla_map --file
New_san-francisco_california.osm.json
```

4. Data Overview

In this section I will talk about some statistics of my data.

- File Sizes:
  san-francisco-california.osm ……1.06 GB
  New.san-francisco-california.osm.json ……1.66 GB

- Number of documents:
  > db.final_map.find().count()
  5545841

- Number of Nodes:
  > db.final_map.find({"type" :"node"}).count()
  4959796

- Number of ways:
  > db.final_map.find({"type" :"way"}).count()
  585909

5. Data Exploration with Queries by Pymongo

- Finding 5 Top Cuisines:
  File “cuisines_query.py” was used.
  The written pipeline in this file is:

  ```
  match = {"$match":{"amenity":{"$exists":1},"cuisine":{"$exists":1} ,"amenity":"restaurant"}}
  group = {"$group":{"_id":"$cuisine", "count":{"$sum":1}}}
  ```

```
sort = {"$sort": {"count": -1}}
limit = {"$limit" : 5}

pipeline = [match,group,sort,limit]

{u'_id': u'mexican', u'count': 228}
{u'_id': u'chinese', u'count': 177}
{u'_id': u'pizza', u'count': 172}
{u'_id': u'japanese', u'count': 150}
{u'_id': u'italian', u'count': 144}
```

- Finding 5 top religion:
  File “top_religion.py” was applied.
  The related pipeline is:
  match ={"$match":{"amenity":{"$exists":1},"religion":{"$exists":1},"amenity":"place of worship"}}
  group= {"$group":{"_id":"$religion", "count":{"$sum":1}}}
  sort = {"$sort": {"count": -1}}
  limit = {"$limit" : 5}
  pipeline = [match,group, sort, limit]

  {u'_id': u'christian', u'count': 1008}
  {u'_id': u'buddhist', u'count': 34}
  {u'_id': u'jewish', u'count': 19}
  {u'_id': u'muslim', u'count': 8}
  {u'_id': u'taoist', u'count': 3}

- Finding 5 top amenities:
  The query was generated with “top_amenities.py”.
  The applied pipeline is:
  match = {"$match": {"amenity":{"$exists":1}}}
  group = {"$group":{"_id":"$amenity","count":{"$sum":1}}}
  sort = {"$sort":{"count": -1}}
  limit = {"$limit" : 5}

  pipeline = [match,group,sort,limit]

  {u'_id': u'parking', u'count': 4520}
  {u'_id': u'restaurant', u'count': 3266}
  {u'_id': u'school', u'count': 1294}
  {u'_id': u'bench', u'count': 1179}

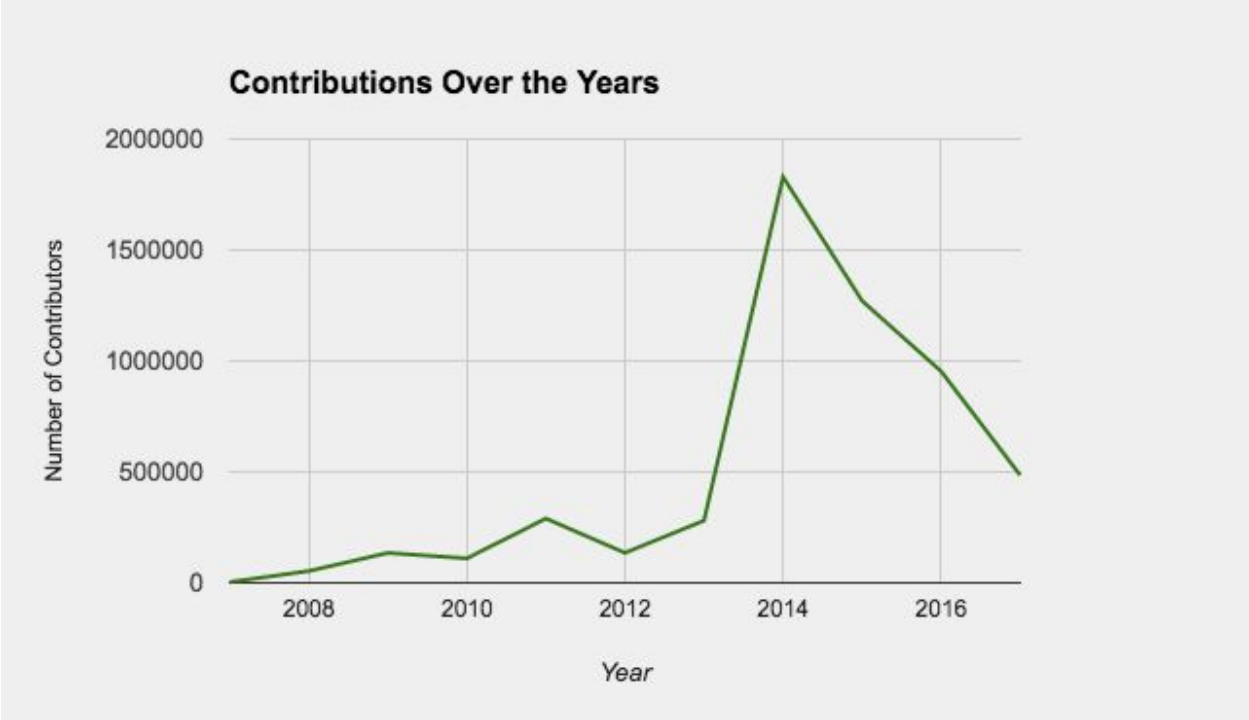
{u'_id': u'place of worship', u'count': 1156}

6. Additional Ideas: Change in Contributors' Rate Over the Years and new Idea

- In order to map the contributor's behaviour, First, I started to find out how many contributions were done in different years. To do so, I used the "date_query.py".

The applied pipeline is:

```
match = {"$match": {"created.timestamp.year":{"$exists":1}}}
group1 = {"$group":{
        "_id" :  "$created.timestamp.year",
        "count" : {"$sum" : 1}
   }}
pipeline = [match, group1]
```



{u'_id': 2007, u'count': 3586}
{u'_id': 2008, u'count': 53256}
{u'_id': 2009, u'count': 134816}
{u'_id': 2010, u'count': 110224}
{u'_id': 2011, u'count': 289873}
{u'_id': 2012, u'count': 135161}
{u'_id': 2013, u'count': 280649}

{u'_id': 2014, u'count': 1829080}
{u'_id': 2015, u'count': 1270938}
{u'_id': 2016, u'count': 953160}
{u'_id': 2017, u'count': 485098}

As one may notice, the highest contribution is related to 2014 with 1,829,080 contributions.

In the next step, I found the top 5 contributors. I used “users_query.py” and the applied pipeline is:

```
match = {"$match": {"created.timestamp.year":{"$exists":1}}}
group={"$group":{"_id":{"year":"$created.timestamp.year","User":"$created.user"},"count":{"$sum":1}}}
 sort = {"$sort":{"count": -1}}
  limit = {"$limit" : 5}

    pipeline = [match,group,sort,limit]
```

{u'_id': {u'User': u'ediyes', u'year': 2014}, u'count': 684050}
{u'_id': {u'User': u'Luis36995', u'year': 2014}, u'count': 541228}
{u'_id': {u'User': u'Rub21', u'year': 2014}, u'count': 390723}
{u'_id': {u'User': u'ediyes', u'year': 2015}, u'count': 161303}
{u'_id': {u'User': u'RichRico', u'year': 2017}, u'count': 137808}

Then using “top_user.py”, for each of these top users, I found their contribution as follow:
ediyes:
{u'_id': 2014, u'count': 684050}
{u'_id': 2015, u'count': 161303}
{u'_id': 2016, u'count': 73419}
{u'_id': 2017, u'count': 98}

Luis36995:
{u'_id': 2014, u'count': 541228}
{u'_id': 2015, u'count': 124173}
{u'_id': 2016, u'count': 44290}
{u'_id': 2017, u'count': 724}

Rub21:
{u'_id': 2012, u'count': 49}

{u'_id': 2013, u'count': 15}
{u'_id': 2014, u'count': 390723}
{u'_id': 2015, u'count': 4095}
{u'_id': 2016, u'count': 150}

RichRico:
{u'_id': 2017, u'count': 137808}
{u'_id': 2016, u'count': 125250}
{u'_id': 2015, u'count': 98930}
{u'_id': 2014, u'count': 4}

- New Additional Idea

After exploring the data, e.g. the number of contributors over the years, in Section 5 and also this section, it came to me that based on the growth rate of contributors in these years, we are ultimately able to have plenty of useful information about every point in the US (and perhaps some other countries).

Therefore, my interest is to build a recommendation system for new retail businesses. For example, if one is interested in opening a store, e.g. a restaurant serving Indian cuisine, one may wonder where to open in order to maximize the revenue and perhaps at the same time reduce cost of advertising, etc. Integrating OSM data with other financial details, one can rank the possibilities, for example having such cuisine in the neighborhood close to a Buddhism worship place, or a middle-eastern cuisine close to a Mosque, etc. This system which requires accurate prediction system, can take advantage of the available data to make the relevant recommendations. Speaking of implementing an application one can also incorporate the income information in each neighborhood (collecting data from the relevant sources) to make a more detailed prediction.

Conclusion

San Francisco dataset which was relatively large (larger than 1 GB) contained some issues that had to be fixed. Therefore in order to the related analysis first I had to perform some data cleaning steps. Doing so multiple analysis have led to various interesting observation regarding San Francisco. For example the OpenStreetMap as expected, the main cuisine type was mexican and there is a huge different in the number of worship
places for different religions in the city.
As the final take away point, one can see that the number of contributors has been fluctuating over the time whereas the maximum occurred in 2014. Hopefully with an improvement user contributions we will have more accurate data for the years to come.