P3 Project: Wrangle OpenStreetMap Data

Kristofer Maanum

Map Area: Singapore

MapZen: Metro Extract, OSM Data for Singapore

1. Problems Encountered in the Map

Upon review of the data a few problems were identified. The main problems addressed were:

1. Over-abbreviated street names (with some non-English abbreviations).
2. Non-standard set of data fields
3. Multiple names for a single location with the main name sometimes not in Latin characters

Abbreviations in Street Names

Review of the data showed that many abbreviations were used and they weren't always consistent. Additionally, some non-English abbreviations were used that could be confusing to an English-speaking audience (e.g. "Jln" for "Jalan"). Abbreviations were removed to the extent possible.

Non-standard set of data fields

The OSM format allows users to enter any data about a location in the "tag" tag with the "k" and "v" attributes representing the key and value, respectively. Including all these values without review could make the Mongo DB documents messy and the database bloated. However, these values may have important information that we might use at a later time.

To address this, all "tag" information that was not specifically handled was moved under a new field "addl_data". In this way the data would be retained but could easily be dropped at a later date if so desired.

Name Handling

Multiple names could be listed for a single location. Additionally, some locations used non-Latin characters for their main name (rendering the name unreadable for some audiences). To help address this a single name was selected for the "name" field with all remaining names moved under field "alt_names".

Data Schema

Based off the work in the problem steps and the handling of additional data and names listed above, the resultant JSON data schema is as follows (indicative only):

```
{
  "name": "",
  "id": "",
  "node_refs": [],
  "created": {
    "changeset": "",
    "user": "",
    "version": "",
    "uid": "",
    "timestamp": ""
  },
  "addr": {
    "country": "",
    "city": "",
    "housenumber": "",
    "postcode": "",
    "street": ""
  },
  "addl_info": {},
  "alt_names": {},
  "type": "way"
}
```

A sample entry is listed below:

OSM XML Sample	**Resultant JSON**

<way changeset="27836138" id="319830324"
timestamp="2015-01-01T10:39:26Z" uid="838520"
user="Pizza1016" version="1">
        <nd ref="1548423269" />
        <nd ref="1548423272" />
        <tag k="lit" v="no" />
        <tag k="ref" v="E2" />
        <tag k="foot" v="no" />
        <tag k="name" v="Lebuhraya Utara-Selatan"/>
        <tag k="horse" v="no" />
        <tag k="lanes" v="2" />
        <tag k="layer" v="1" />
        <tag k="access" v="yes" />
        <tag k="bridge" v="yes" />
        <tag k="oneway" v="yes" />
        <tag k="bicycle" v="no" />
        <tag k="highway" v="motorway" />
        <tag k="name:en" v="North-South Expressway"/>
        <tag k="name:zh" v="南北大道" />
        <tag k="surface" v="asphalt" />
        <tag k="cycleway" v="no" />
        <tag k="maxspeed" v="110" />
        <tag k="sidewalk" v="none" />
        <tag k="wikipedia" v="en:North-South
Expressway Southern Route" />
        <tag k="destination" v="Sungai Besi" />
        <tag k="motor_vehicle" v="yes" />
</way>
	{
	  "name": "Lebuhraya Utara-Selatan",
	  "id": "319830324",
	  "node_refs": [
	    "1548423269",
	    "1548423272"
	  ],
	  "created": {
	    "changeset": "27836138",
	    "user": "Pizza1016",
	    "version": "1",
	    "uid": "838520",
	    "timestamp": "2015-01-01T10:39:26Z"
	  },
	  "addl_info": {
	    "cycleway": "no",
	    "wikipedia": "en:North-South
	Expressway Southern Route",
	    "oneway": "yes",
	    "sidewalk": "none",
	    "horse": "no",
	    "maxspeed": "110",
	    "bridge": "yes",
	    "lanes": "2",
	    "lit": "no",
	    "bicycle": "no",

```
    "destination": "Sungai Besi",
    "ref": "E2",
    "foot": "no",
    "motor_vehicle": "yes",
    "access": "yes",
    "layer": "1",
    "highway": "motorway",
    "surface": "asphalt"
  },
  "alt_names": {
    "name:en": "North-South Expressway",
    "name:zh": "南北大道"
  },
  "type": "way"
}
```

2. Data Overview

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

File sizes

```
singapore.osm ......... 206.7 MB
singapore.osm.json .... 296.4 MB
```

Number of documents

Command	**Result**
`db.char.find().count()`	1086522

Number of nodes

Command	**Result**
`db.char.find({"type":"node"}).count()`	942692

Number of ways

Command	**Result**
`db.char.find({"type":"way"}).count()`	`142057`

Number of Unique Users

Command	**Result**
`db.SG.distinct("created.user").length`	`1172`

Top 1 contributing user

Command	**Result**

`db.SG.aggregate([`
       `{ $group : {_id :'$created.user',`
              `count : { $sum : 1 }}},`
       `{ $sort : { count : -1 }},`
       `{ $limit : 1 }`
`])`
	`{ "_id" : "JaLooNz", "count" : 300416}`

Top 1 contributing user

Command	**Result**

`db.SG.aggregate([`
       `{ $group : {_id :'$created.user',`
              `count : { $sum : 1 }}},`
       `{ $sort : { count : -1 }},`
       `{ $limit : 1 }`
`])`
	`{ "_id" : "JaLooNz", "count" : 300416}`

Number of users appearing only once (having 1 post)

Command	Result

`db.SG.aggregate([`
`      { $group : { _id:'$created.user',`
`            count:{ $sum : 1 }}},`
`      { $group : { _id : '$count' ,`
`            num_users : { $sum : 1 }}},`
`      { $sort : { _id : 1 }},`
`      { $limit : 1 }`
`])`
	`{"_id":1,"num_users":223}`

3. Additional Ideas

Data Entry Standardization

There is a wide range of what data is included for each entry in the database. This seems to be due to the flexibility of the “tag” tags with “k” and “v” attributes. When using the OpenStreetMap editing front-end, the editor is prompted with main tags that might apply to the map object in general. Additionally, the editor can create and add values to their own tags. This can lead to some clear deficiencies in the data:

Restaurants with indicated ‘Cuisine’

Command	Result

`db.SG.aggregate([`
`    { $match : {`
`        'addl_info.amenity':'restaurant'}},`
`    { $group : {`
`        _id : '$addl_info.cuisine',`
	`{ "_id" : null, "count" : 681 }`
	`{ "_id" : "chinese", "count" : 90 }`
	`{ "_id" : "japanese", "count" : 32 }`
	`{ "_id" : "korean", "count" : 31 }`
	`{ "_id" : "italian", "count" : 29 }`
	`{ "_id" : "pizza", "count" : 26 }`

```
        count : { $sum : 1 }}},
    { $sort : { count : -1 }},
    { $limit : 10 }
])
```

```
{ "_id" : "indian", "count" : 23 }
{ "_id" : "asian", "count" : 23 }
{ "_id" : "regional", "count" : 15 }
{ "_id" : "french", "count" : 13 }
```

In this result we can see that most restaurants do not have a cuisine indicated. Additionally, some cuisine descriptions seem to overlap (e.g. ‘Korean’ might also be ‘Regional’ and is a subset of ‘Asian’).

Place of Worship with indicated ‘Religion’

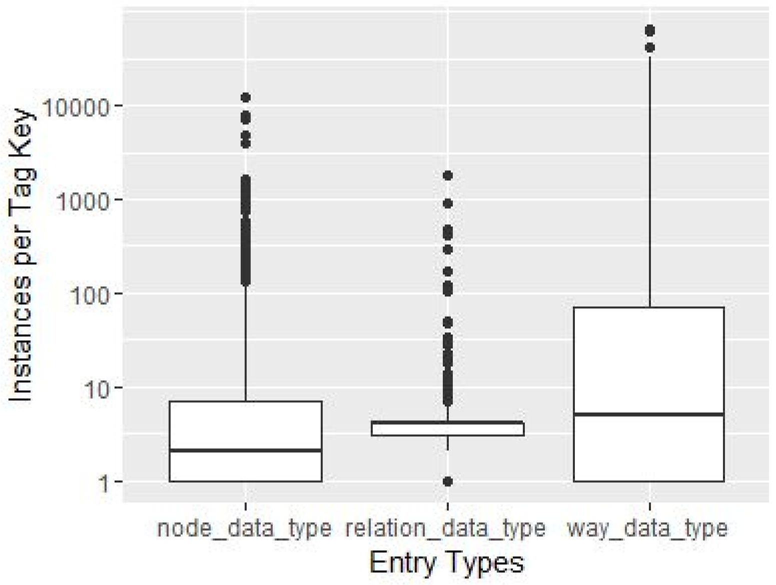
Command

```
db.SG.aggregate([
    { $match :
        {'addl_info.amenity':
            'place_of_worship'}},
    { $group :
        { _id : '$addl_info.religion',
        count : { $sum : 1 }}},
    { $sort : { count : -1 }},
    { $limit : 10 }
])
```

Result

```
{ "_id" : "muslim", "count" : 464 }
{ "_id" : "christian", "count" : 182 }
{ "_id" : null, "count" : 95 }
{ "_id" : "buddhist", "count" : 65 }
{ "_id" : "hindu", "count" : 16 }
{ "_id" : "taoist", "count" : 7 }
{ "_id" : "jewish", "count" : 4 }
{ "_id" : "sikh", "count" : 3 }
```

Similar to the lack of ‘cuisine’ for restaurants, there are many places of worship that do not have a religion indicated.



This boxplot shows the count of each tag key per entry type. Here we can see that there are many tags which appear only a handful of times in the database. There could be a lot of overlap between meanings in these tags.

To help address these issues some additional standardization could be promoted or enforced within OpenStreetMap.

With enforcement of a standard there would be tradeoff between flexibility and full indexed fields however. For example, if one needs to enter a ‘cuisine’ for a restaurant that serves Thai, Chinese and Burmese dishes, perhaps ‘Asian’ is more appropriate than a more specific term. Allowing for more than one ‘cuisine’ could address this type of issue however.

Instead of enforcement, a standard could also be promoted. The data entry front end could be improved to provide suggestions when a new value is entered. As examples:

1. **Suggest existing values**: if a user enters “Thai Food” which does not already exist for the “cuisine” tag, the system could suggest the user to use “Thai” instead. Another example
2. **Auto-fill addresses**: If a location is added or edited within a certain city/country, the system could complete the city/country tags and prompt the user to verify the values.
3. **Verify known formats:** In some cases, a data format is already established. For example, Singapore uses 6-digit postal codes. These formats could be enforced in the system upon data entry/editing (e.g. if “Country” is “Singapore” then “Postal Code” will be validated to 6 digits)

Additional data exploration using MongoDB queries

Contributions by year

Command	**Result**

`db.SG.aggregate([`
`        { $group : { _id:'$created.user',`
`                count : { $sum : 1 }}},`
`        { $sort : { count : -1 }},`
`        { $limit : 1 }`
`])`
	`{ "_id" : "2007", "count" : 311 }`
	`{ "_id" : "2008", "count" : 8162 }`
	`{ "_id" : "2009", "count" : 11560 }`
	`{ "_id" : "2010", "count" : 52875 }`
	`{ "_id" : "2011", "count" : 119163 }`
	`{ "_id" : "2012", "count" : 293408 }`
	`{ "_id" : "2013", "count" : 157375 }`
	`{ "_id" : "2014", "count" : 222880 }`

	`{ "_id" : "2015", "count" : 188018 }`
	`{ "_id" : "2016", "count" : 32770 }`

Top 10 Cuisines Listed

Command	Result

`db.SG.aggregate([`
`   {$match : {'addl_info.cuisine':`
`      { $exists : true, $ne : null }}},`
`   {$group : {_id:'$addl_info.cuisine',`
`      count : { $sum : 1 }}},`
`   { $sort : { count : -1 }},`
`   { $limit : 10 }`
`])`
	`{ "_id" : "chinese", "count" : 90 }`
	`{ "_id" : "burger", "count" : 63 }`
	`{ "_id" : "coffee_shop", "count" : 37 }`
	`{ "_id" : "chicken", "count" : 36 }`
	`{ "_id" : "japanese", "count" : 33 }`
	`{ "_id" : "korean", "count" : 31 }`
	`{ "_id" : "italian", "count" : 29 }`
	`{ "_id" : "asian", "count" : 28 }`
	`{ "_id" : "pizza", "count" : 27 }`
	`{ "_id" : "indian", "count" : 24 }`

Top 10 Religions Listed

Command	Result

`db.SG.aggregate([`
`  { $match : {'addl_info.religion':`
`    { $exists : true, $ne : null }}},`
`  { $group : {_id:'$addl_info.religion',`
`    count : { $sum : 1 }}},`
`  { $sort : { count : -1 }},`
`  { $limit : 10 }`
`])`
	`{ "_id" : "muslim", "count" : 471 }`
	`{ "_id" : "christian", "count" : 185 }`
	`{ "_id" : "buddhist", "count" : 67 }`
	`{ "_id" : "hindu", "count" : 16 }`
	`{ "_id" : "taoist", "count" : 7 }`
	`{ "_id" : "jewish", "count" : 4 }`
	`{ "_id" : "sikh", "count" : 3 }`
	`{ "_id" : "shinto", "count" : 1 }`

Conclusion

It's clear that the data encompasses the general area around Singapore, including some Malaysian and Indonesian locations. The data structure was consistent and was further formalized before loading into Mongo DB. However, the open nature of the data entry and the flexible tags ("k" and "v") allowed a wide range of data to be included (or not included), making it difficult to pull any real insights or to assess the completeness of the data.