OpenStreetMap Data Case Study

Map Area

I decided to explore and edit the osm data of Hamburg in Germany. It is the city I live and work in and would like to contribute to. The osm extract was downloaded via mapzen.

Problems Encountered in the Data

The data donwloaded was passed through the subset creation code and elements were filtered for only those that contain `'node', 'way', 'relation'`. When running the auditing the remaining data, only a few problems came up:

- Some elements did not contain a user name or uid
- Phone numbers were not formatted in a unified way
- Websites were not formatted in a unified way

The ZIP codes all looked fine, the street names were correct and even the house numbers.

Missing user name and uid

There were 3 elements where uid and user were missing which caused a validation error and needed to be replaced. As mentioned in the Udacity forums, one solution is to replace those missing values with `0000` by using this code:

```
    if element.tag == 'node':
        for node in NODE_FIELDS:
            try:
                node_attribs[node] = element.attrib[node]
            except:
                node_attribs[node] = '00000'
                pass
```

Phone numbers

There is a inconsistency in the phone number format which could have been expected as there are many different ways phone numbers are formatted in Germany. When looking at the most used phone numbers with this query:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
          UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='phone'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 100;
```

which results in the follwing table:

value	count
112	8
+49 4182 21172	7
+49 40 74100	4
+49 800 0282266	3
Anbieter möchte nicht veröffentlicht werden	3
+49 4101 375744	2
+49 4131 187458	2
+49 4181 3700	2
+49 4182 501842	2
+49 4186 892491	2

I started creating regex replacements but soon realized that the it would not make sense to replace eg. the 0 with + in the beginning as there are cases such as `112` being the number emergency number. Also some phone numbers contained useful information for the OSM community such as `Anbieter möchte nicht veröffentlicht werden` meaning that the owner of the business did not want his number to be published in OSM. So I decided to leave the phone numbers as they are and handle them as strings.

Surprisingly Clean Data

Street Names

Once the data was imported to SQL, some basic querying did not reveal any suspicious data. Neiter looking at the upper, lower or middle values revealed obvious mistakes. Someone seems to have done some good data cleaning

before me here, which is very good because the street names do not follow an easy to parse schema as it can be seen in the extract below:

Example of least occuring streets that only had 1 entry:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
          UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='street'
GROUP BY tags.value
ORDER BY count ASC
LIMIT 100;
```

value	count
Aldermannweg	1
Alemannen Weg	1
Alfred-Johann-Levy-Straße	1
Allersstraße	1
Allhornstieg	1
Almtweg	1
Alsterkrüger Kehre	1
Alsterring	1
Alt Moorburg	1
Altdammstücken	1

Postal Codes

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
          UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
GROUP BY tags.value
ORDER BY count DESC;
```

Here are the top ten results, beginning with the highest count:

postal code	count

21244	2756
21335	1293
21465	1286
21255	1171
22393	1167
21337	1119
25335	1084
24558	1016
21035	999
21339	994

All of the postal codes are 5 digits long and start with a two (`2xxxx`) which is the format that should be expected for postcodes in this area.

Cities

Sort cities by count, descending

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city'
GROUP BY tags.value
ORDER BY count DESC;
```

Number of unique cities in the area

```
SELECT COUNT(distinct(value)) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city';
```

And, the first 10 of all 425 results:

value	count
Hamburg	40655
Lüneburg	3408

value	count
Elmshorn	1820
Buchholz in der Nordheide	1685
Norderstedt	1562
Stade	1175
Reinbek	1055
Henstedt-Ulzburg	1012
Bad Oldesloe	883
Tostedt	801

Which is mostly because there are a lot of small villages near to Hamburg.

Data Overview and Additional Ideas

This section contains basic statistics about the dataset, the MongoDB queries used to gather them, and some additional ideas about the data in context.

File sizes

```
hamburg_sample_k5.osm ... 294 MB
charlotte.db ............ 129 MB
nodes.csv ............... 90.8 MB
nodes_tags.csv .......... 8.5 MB
ways.csv ................ 12 MB
ways_tags.csv ........... 21.9 MB
ways_nodes.cv ........... 35 MB
```

Number of nodes

```
SELECT COUNT(*) FROM nodes;
```

1131856

Number of ways

```
SELECT COUNT(*) FROM ways;
```

207498

Number of unique users

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

3773

This is a very high number of participants in comparison to the sample submission provided by Udacity – especially when calculating ratios such as ~300 nodes per user.

Number of users appearing only once (having 1 post)

```
SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
     GROUP BY e.user
     HAVING num=1)  u;
```

924

Even though this is again a high number, that also means that 2849 users or 75% posted data more than once.

Additional Ideas

Contributor statistics

The contributions is very equal, and the dataset suggest that people spent a lot of time cleaning the data with great carefuness. This could be leveraged by introducing a live leaderboard for the Hamburg area, as it is not hard to move up in the ladder given the equal distribution of submissions.

```
SELECT e.user, COUNT(*) as num, round(COUNT(*)/1131856.0,4)*100 as perc
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 20;
```

user	number of nodes	perc_of_total

smarties	70297	6.21%
svbr	61719	5.45%
OSchlüter	49935	4.41%
cassy	49767	4.4%
jbhh12	47201	4.17%
4Ems	45294	4.0%
fahrrad	37522	3.32%
sundew	36848	3.26%
streele	35672	3.15%
Abendstund	35187	3.11%
glühwürmchen	26704	2.36%
Divjo	23815	2.1%
vademecum	23294	2.06%
uebi64	21857	1.93%
Kohlmeise	21113	1.87%
simlox	20865	1.84%
Omegatherion	19346	1.71%
Swen Wacker	19327	1.71%
HoMiko	17380	1.54%
westnordost	16088	1.42%

Privacy tags

The privacy concerns in Germany is quiet big which could be seen in the website and phone data above. To be able to handle those tags it would be good to have a standardized value for all those keys where people did not want their data in osm. This could possibly also be done in the underlying data structure by introducing a privacy tag/attribute in the XML file. But changing the underlying data structure would add additional complexity and also not a need that needs to be adressed in OSM in general. So probably a value could be defined for those values that were deleted due to privacy concerns (eg. `deleted_privacy`).

Additional Data Exploration

Top 10 appearing amenities

```
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

amenity	count
bench	1250
post_box	653
restaurant	509
parking	451
recycling	422
vending_machine	320
bicycle_parking	264
waste_basket	244
fast_food	224
cafe	206

This is very funny, people in Hamburg seem to be enjoying sitting on benches even more than going to restaurants. On the other hand this might also be due to the lack of information provided about those amenities in the top 10 in other, commercial platforms such as Google, Yelp or Foursquare.

Most popular cuisines

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

cuisine	count
italian	69
greek	33
german	32
chinese	19
regional	17
asian	15
indian	9
burger	8
portuguese	7
steak_house	7

Conclusion

The OSM data was suprisingly accurate and complete. It seems like Hamburg has been taken care of quiet a lot. It is the second largest city in Germany and regulary hosts the CCC conference which has always also been about privacy which might be one of the reasons for the huge amount of unique users as well as the data quality and values deleted due to privacy concerns. All in all it seems that restaurants and other amenities apart from benches and waste baskets are not too prominent in OSM, which could be fixed by importing data from Google&Yelp into OSM (programmatically).