OpenStreetMap Data Case Study

Map Area

I chose to look at the OpenStreetMap (OSM) data from Düsseldorf, in the very west of Germany. I was raised in the city and moved back to the city nine years ago. Fun fact: Düsseldorf is considered as a city with one of the highest living standards in the world (https://www.mercer.com/newsroom/2017-quality-of-living-survey.html) in the world. I downloaded the already prepared file from https://mapzen.com/data/metro-extracts/metro/duesseldorf_germany/.

Positive Surprises and Problems Encountered in the Map

Positive Surprises

Postal codes are fine

Inconsistent postal codes didn't seem to be a problem as it was in the sample project. The German postal code system is very straight forward: always exactly five digits. Even the simple regex search in the Atom editor – \s[0-9]{5}\ and [0-9]{5}\s – to check if any postal code has been injected with an unintentional white space didn't return any results. This must be due to German accuracy and that German postal code system is very straight forward: it's always five digits. Letters are not allowed.

The Spelling of the Word "Straße"

Straße means Street in German. Obviously, one of the most important words in addresses. As Straße is written with a very German particular letter, the grapheme ***ß***, I wondered if some preferred to write the grapheme's substitute: the ***ss***. However, this was not the case. ***SS*** was solely used for email addresses and URLs.

Problems

Problems were encountered with phone and house numbers. This function was used to print them to gain an overview:

```
def audit(osmfile):
    osm_file = open(osmfile, "r")
    phone_numbers = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if tag.attrib['k'] == "phone":
                    #print phone numbers as entered, then the cleaned version
                    print tag.attrib['v']
                    print clean_phone(tag.attrib['v'])
                # uncomment if you want to see the housenumers
                #   if tag.attrib['k'] == "addr:housenumber":
```

```
                #       print tag.attrib['v']
    osm_file.close()
    return phone_numbers
  ```


Phone numbers

As there’s no single standard here, phone numbers were entered in multiple ways:

- +49 211 123456
- +49 211 / 123456
- +49211123456
- 0049 211 123456
- +49 (211) 123456
- 021113456
- 0211 123456
- +49 (0)211 123456
- +49-211-123456

This function was written to mitigate the issue:

```python
def clean_phone(phone_numbers):
    clean_number = re.sub(r'^00','+', phone_numbers)
    clean_number = re.sub(r'-',' ', clean_number)
    clean_number = re.sub(r'\/\s','', clean_number)
    clean_number = re.sub(r'\(0\)','', clean_number)
    clean_number = re.sub(r'^0211','+49 211', clean_number)
    return clean_number
  ```

House numbers

House numbers stretching more than a single number (e.g., 22-24) have been entered differently, such as:

- 113-119
- 21,23
- 29; 31

However, most numbers are in the first format and should be the stated as the correct form.

Data Overview

All queries can be found in the query.py file. This is the output of the file:

File sizes

duesseldorf.osm.....: 662M dussosm.db.....: 390M
nodes.csv.....: 212M nodes_tags.csv.....: 23M
sample.osm.....: 6M

ways.csv.....: 29M
ways_nodes.csv.....: 91M
ways_tags.csv.....: 64M

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

```
2760081
```

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

```
number of ways 523867
```

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

```
number of contributors (2292,)
```

```
SELECT user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
GROUP BY user
ORDER BY num DESC LIMIT 10;
```

```
    (u'black_bike', 585316)
    (u'Antikalk', 442759)
    (u'Zyras', 230609)
    (u'mighty_eighty', 180603)
    (u'WoGraSo', 173613)
    (u'EinKonstanzer', 173138)
    (u'rurseekatze', 158304)
    (u'rabenkind', 135121)
    (u'Athemis', 133331)
    (u'Kettwicht', 99781)
```

```
SELECT sub.value, COUNT(*) as num
FROM (SELECT key, value FROM nodes_tags UNION ALL SELECT key, value FROM ways_tags)
sub
WHERE sub.key = "amenity"
GROUP BY sub.value
ORDER BY num DESC LIMIT 10;
```

```
    (u'parking', 5573)
    (u'bench', 2934)
```

```
    (u'recycling', 1358)
    (u'restaurant', 1187)
    (u'waste_basket', 1107)
    (u'post_box', 803)
    (u'bicycle_parking', 718)
    (u'shelter', 686)
    (u'vending_machine', 685)
    (u'fast_food', 576)
```

```
SELECT sub.value, COUNT(*) as num
FROM (SELECT key, value FROM nodes_tags UNION ALL SELECT key, value FROM ways_tags)
sub
WHERE sub.key = "suburb" GROUP BY sub.value ORDER BY num DESC LIMIT 10;
```

```
    (u'Strümp', 1703)
    (u'Wersten', 1012)
    (u'Bilk', 993)
    (u'Gerresheim', 989)
    (u'Unterrath', 793)
    (u'Benrath', 770)
    (u'Pempelfort', 732)
    (u'Eller', 725)
    (u'Düsseltal', 700)
    (u'Oberkassel', 680)
```

Dataset Improvement

Consistency

For this project, I just corrected some erroneous ways of wrongly formatted phone numbers. However, people have entered the phone numbers in even more different ways (s. above). So the function needs to be extended to all use cases. Maybe, OSM could provide a strict mask/form how to enter phone numbers. The same goes for house numbers regarding being saved consistently.

Benefits are: - As some number entries are lacking the international code (+49), foreign users cannot simply call the number directly by just clicking on it. With the strict format which includes +49, this issue would be passé. - Consistency makes the life easier for: - data analysts to analyze the phone/house numbers for statistics (e.g. which numbers are directly from Düsseldorf +49 211 vs. Neuss + 49 02131) - users to search for numbers

Potential issues are: - Contributors might find it annoying to type numbers in a fixed format. They might just want to quickly copy and paste the number from somewhere else. So the risk is that no number is going to be entered at all.

More contribution

Having a background in the insurance industry, many insurance companies are more and more communicating with (potential) customers when they reach a specific spot or location (e.g.

customer is 2000 m above sea level, suggesting an accident insurance) They call it situative offers.

Google maps is going a similar way when they ask questions about a location when users enter restaurants or cafés. Like Google, OSM could combine the situative approach with some gamification.

An idea would be, whenever a contributor enters a to OSM known amenity, missing information are asked. Contributor receive points, badges and can compare themselves in a regional leaderboard.

Benefits are: - Missing information are added faster. Contribution and thus completion of the map rises - Simply more fun to contribute

Potential issues are: - Contributors might find this an intrusion to their privacy - The risk of a *crowding-out effect* of the intrinsic motivation to contribution due the feeling of competition

Conclusion

The data from Düsseldorf is generally in a good shape. However, more consistency and contribution is always beneficial.

Resources

- https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md
- https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f
- Udacity Forum
- Udacity 1:1 appointment
- Udacity Reviewer