OpenStreetMap Data Case Study

Problems encountered in the map

Initially I downloaded a small sample of the Ho Chi Minh City area, and then running it against a provisional data.py file. Later on, after importing the data into SQL, I found several problems with the data. Below is some of them.

- k and value don't match in meaning
- Inconsistency in using k='street' and k='route' to describe the street name
- Inconsistency in using k='name' to convey street name, while having k='street'
- Duplicated values between k='name' and k='name:vi' because they are for the most part both in Vietnamese

Due to the limitation of time, I only cleaned 2 of the above problems, which are the inconsistency problems.

Street and Route Inconsistency

As mentioned above, the dataset has k as street and route to both describe street names, so this will create inconsistency when later on we perform analysis on it. Therefore, I wrote a small function in audit.py to make sure that my observation in SQL about this problem was right, and after running it in audit.py, I used my advantage in Vietnamese to confirm again that they were both describing the same thing. Then, I wrote a small function call update_route in data.py to fix the problem before importing the data into csv files.

```
def update_route(tag):
    tag.attrib['k'] = 'street'
```

After updating, all the k='route' will be turned into k='street'

Street and Name Inconsistency

As above, I used a function in audit.py to confirm my observation about the mixture of actual names and street names. Again, I used Vietnamese as an advantage to clean this inconsistency. In Vietnamese, we mostly only have 2 ways to call street, which are "Đường" and "Hẻm" at the beginning of a phrase. And I audited the data using audit.py to see if there's any other way. There seemed none, but I got to collect variations of “Đường" and “Hẻm", and used them in my update function to clean the data. Therefore, I went ahead and used the function below in data.py to clean the data before importing to a csv file.

```
expected = ["Duong","duong", "đường", "Đường", 'DUONG', "Hẻm", "hẻm",'Hem','HẺM']

def is_street(string):
    for i in expected:
        if i in string:
            return True

def update_street(tag):
    if is_street(tag.attrib['v']):
        tag.attrib['k'] = 'street'
```

After running this function, any k='name' that had a variation of "Hẻm" or "Đường" would be turned into k='street'

Overview of the data

This section contains basic statistics about the dataset, and I used SQL queries in jupyter notebook to gather them, except for the file size section.

File size

```
hcm.db  = 43.4 MB
hcm.osm  = 84.1 MB
sample.osm  = 1.2 MB
nodes.csv  = 32.0 MB
nodes_tags.csv  = 3.6 MB
ways.csv  = 2.7 MB
ways_nodes.csv  = 10.6 MB
ways_tags.csv  = 3.4 MB
```

Number of nodes

```
node = cur.execute('SELECT COUNT(*) FROM nodes')
node.fetchone()[0]
```

```
404079
```

Number of ways

```
ways = cur.execute('SELECT COUNT(*) FROM ways')
ways.fetchone()[0]
```

```
47878
```

Number of unique users

```
unique_user = cur.execute('SELECT COUNT(DISTINCT(sq.uid)) \
            FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) as sq')
unique_user.fetchone()[0]
```

```
1596
```

Top 10 contributing users

```
top_10_active = cur.execute('SELECT sq.user, COUNT(*) as num \
            FROM  (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as sq\
            GROUP BY sq.user \
            ORDER BY num DESC \
            LIMIT 10')
top_10_active.fetchall()
```

```
[('TuanIfan', 126416),
 ('Ivan Garcia', 74020),
 ('Dymo12', 38772),
 ('QuangDBui@TMA', 27482),
 ('KimChinhTri', 23062),
 ('528491', 18767),
 ('guenter', 10013),
 ('tandung', 9106),
 ('mrjoba', 7725),
 ('dotnam', 6090)]
```

Number of users appearing only once

```
user_once_only = cur.execute('SELECT count(*)\
            FROM (SELECT sq.user, COUNT(*) as num \
            FROM  (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as sq\
            GROUP BY sq.user \
            HAVING num = 1) as sq2')
user_once_only.fetchone()[0]
```

```
713
```

Additional Idea

What I have done above, by no mean, completely cleaned the dataset, but I have seen a lot of rooms to prevent some problems for the data. One of my suggestions is to set up a way for user to fill in phone number in a consistent way. The reason why I said this is because while doing my exploration on the whole dataset using SQL in Jupyter notebook, I've found an big of amount of dirty data in the phone numbers inputted by users.

```
phone=cur.execute('SELECT sq.value\
            FROM (SELECT value FROM nodes_tags\
            WHERE key="phone"\
            UNION ALL\
            SELECT value FROM ways_tags \
            WHERE key="phone") as sq\
            LIMIT 20')
phone.fetchall()
```

```
[('0839307627',),
 ('+84 8 39991612 ',),
 ('8222373',),
 ('8368761',),
 ('8361679',),
 ('8365073',),
 ('39206472',),
 ('38365868',),
 ('38369123',),
 ('38379589',),
 ('+84 838369268 / +84 838361935',),
 ('38375582',),
 ('+84838370447',),
 ('3829 2772',),
 ('+084 0917 685 464',),
 ('+8438231130',),
 ('01252026241',),
 ('+84 8 38272372',),
 ('+84838389523',),
 ('84839105689',)]
```

Using my local knowledge of phone numbers, I know that the mixture of numbers above consists numbers of landline phones and cellphones. Also, there is an inconsistency between using and not using country code (+84).

This is not a hard problem to solve at all if there is a way to help people to input their phone numbers in the same format, and in my opinion, openstreetmap can help users to create such a format. This format will ensure the consistency in phone number input. The possible problem here is that some user may not have English competency to understand the instruction to follow such a format.