Wrangle OpenStreetMap Data

Map Area

Bishkek, Kyrgyzstan

This is map of the city where I was born, so it was interesting to me to see what database querying reveals, and I liked the opportunity to contribute to its improvement on OpenStreetMap.org. Unfortunately, Bishkek wasn't amongst preselected metro areas, so I used MapZen to make custom extract with boudaries 74.5037007989, 42.7787328152, 74.6717525572, 42.9377928738. As extracted area is rectangular, it contains some adjacent to Bishkek areas, so in the following analysis we should keep it in mind. The extract was created on 2016 November 15, at 01:48 AM and could be found here.

Problems Encountered in the Map

After downloading map of the Bishkek area and running audit functions, I noticed several problems with the data, which I will discuss in the following order:

- Inconsistency in language of street names and types: `"Tchui Strasse", "Manas Avenue", "улица Матросова"`
- Overabbreviated street types with inconsistent position: `"ул. Масыралиева", "Асанбай мкр.", "Kiev Str"`
- Street names without street type: `"Ореховая", "Восток-5", "Щорса"`
- Incorrect postal codes: `"12345", "7220082", "772200"`
- Inconsistent phone number formats, `"+996 555 70 70 74", "(312) 890257", "0(312)54-49-25", "325528"`
- Multiple phone numbers in different formats: `"+996 312 596570, +996 312 596494", "0(312)88-14-14 0(556)11-22-33"`
- Incorrect phone numbers: `"+99655804111", "+9963122115"`
- Format of website links doesn't follow best practices of OpenStreetMap: `"grenki.kg", "www.android.kg"`

Streets cleaning

At first, I decided to change street type so the street field had format: `<street name> <full street type>`. For this I implemented function fix_street_type in street.py. This function checks the beginning and the end of raw street value and correct abbreviated or translated street type to their respective mappings. Also this function handle consistency of street type position: it places corrected street type after street name.

To deal with correcting streets without street types function fix_street in street.py does two things:

- tries to get obbreviated street type by street name from offical kyrgyz post site
- gets street type from manually prepared map of 30 existing street names

Postal codes cleaning

As was mentioned above, processed dataset contains data for Bishkek and some adjacent areas. So I didn't require postal code to be valid only for Bishkek. Otherwise, I only checked if it's valid for Kyrgyzstan: it has to be exacly six digits and first two digits should be "72". So here is the postcode validity function:

```
def is_valid_postcode(postcode):
    if len(postcode) != 6 or postcode[:2] != "72":
        return False
    return postcode.isdigit()
```

After look of invalid postcodes we can see that some of them are probably just typos(7220082, 730077), but some are clearly a randomly typed digits(1234, 12345):

```
postcode | count
1234     | 1
11       | 1
772200   | 3
7220082  | 11
1        | 1
12345    | 1
730077   | 2
```

Anyway, I removed all invalid postcodes, because they are useless.

Phones cleaning

Audit of phone numbers showed that there are 200 phones which are not presented in the advised in OpenStreetMap wiki format: `+<country code> <city code> <local number>`. Besides difference created by formatting(like city code in parenthesis, hyphens in local number, additional spaces etc), there are several possible represenations of valid phone number for Bishkek area. This is how, for example, a valid phone +996 312 613238 could be represented:

- 0996 312 613238
- 9960 312 613238
- 0 312 613238
- 312 613238
- 613238

The function fix_format from phone.py changes any of this possible representations of valid phones (which match regex `(996|0|9960|0996)?([0-9]{3})?[0-9]{6}$`) to the phone number in advised format (which match regex `\+996\s[0-9]{3}\s[0-9]{6}(;\s)?)+$`).

The other problem was that some phone numbers are actually several phone numbers, divided by spaces, commas or semicolons. So clean_phone function from phone.py handles this by dividing different phones into list and fixing format of each phone number separately. Then formatted results are joining with separation by semicolons (which is more common for multiple entries in OpenStreetMap data).

Websites cleaning

OpenStreetMap wiki advises to include the scheme (http or https) in website format. So I check if website values follows this advice:

```
website = re.compile(r'https?://[a-z0-9\./]*')

def url_is_good(url):
    return website_re.match(url)
```

For thouse websites, which doesn't have scheme, I add scheme http, because not every website uses https scheme. Some possibilites to add https scheme will be described in the last section.

Overview of Data

This section contains some overview statistics about the dataset and the SQL queries used to gather them.

File sizes

```
Bishkek.osm ..........121.0 MB
Bishkek.db ............ 64.6 MB
nodes.csv ............. 42.0 MB
nodes_tags.csv ........  0.6 MB
ways.csv ..............  6.9 MB
ways_tags.csv ......... 15.2 MB
ways_nodes.cv ......... 10.2 MB
```

Number of nodes

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

```
515172
```

Number of ways

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

```
116417
```

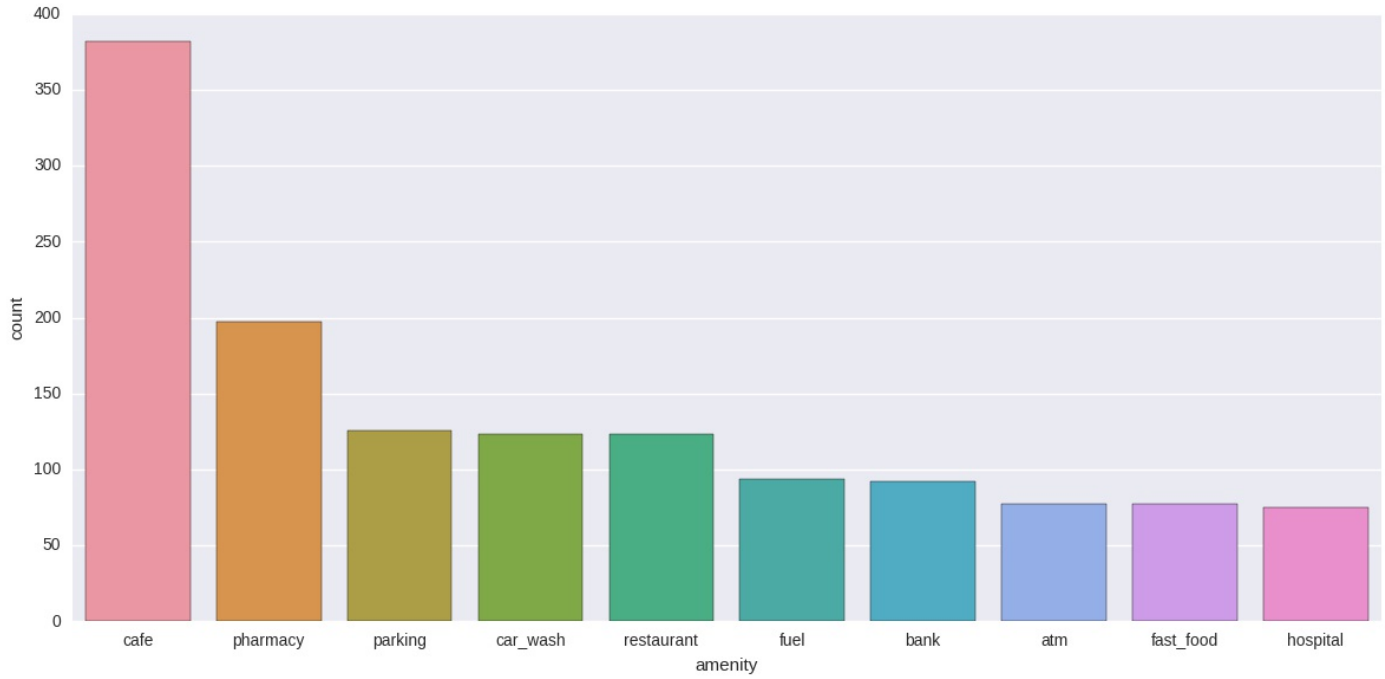
Number of unique users

```
sqlite> SELECT COUNT(*)
FROM (SELECT uid FROM nodes UNION SELECT uid FROM ways);
```

```
687
```

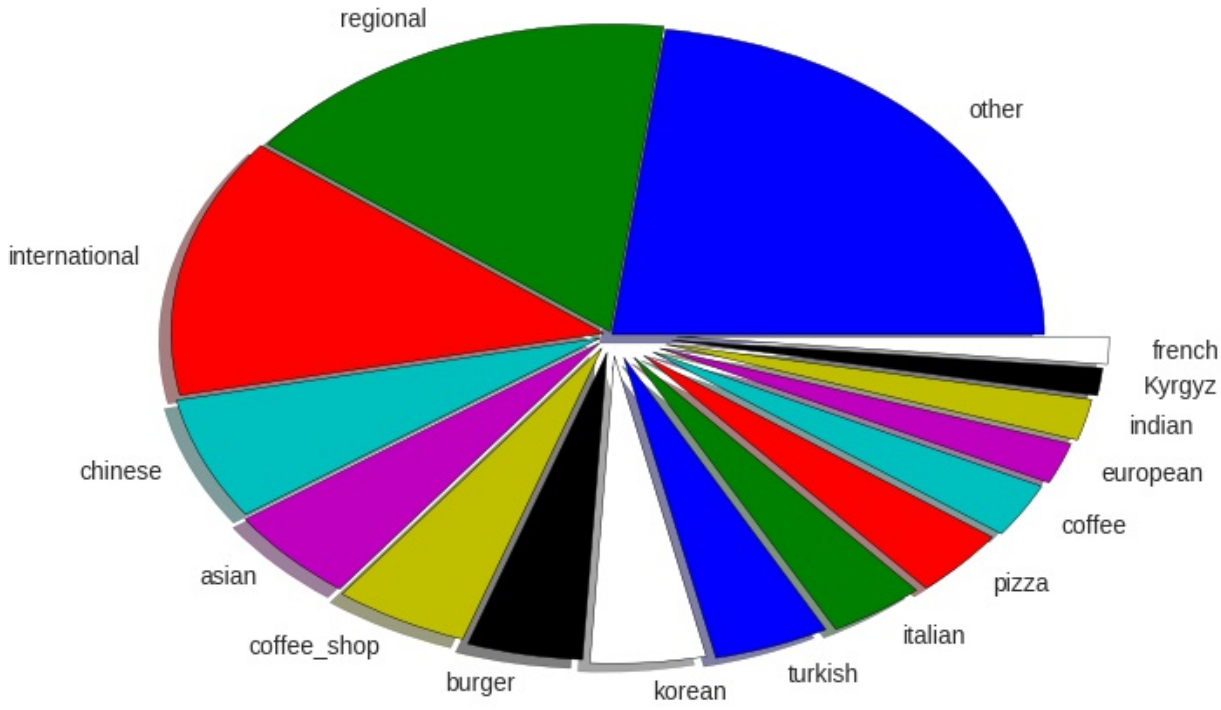
Top 10 amenities

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



Top 15 cuisines

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags WHERE key='cuisine'
GROUP BY value
ORDER BY num DESC LIMIT 15;
```



The top 10 streets with the largest number of associated nodes

```
sqlite> SELECT ways_tags.value, COUNT(ways_nodes.node_id) as num
FROM ways_nodes JOIN ways_tags
WHERE ways_nodes.id=ways_tags.id AND ways_tags.key='street'
GROUP BY ways_nodes.id
ORDER BY num DESC LIMIT 10;
```

```
Тоголок Молдо улица        | 78
Токтогула улица            | 64
Талаа улица                | 60
Профсоюзная улица          | 51
Московская улица           | 47
7 микрорайон               | 44
Эркиндик бульвар           | 43
Ахунбаева Исы улица        | 42
Сыдыкова улица             | 42
Уметалиева Темиркула улица | 41
```

The top 10 nodes with the largest number of corrections

```
>sqlite SELECT nodes_tags.value, nodes.version
FROM nodes JOIN nodes_tags
WHERE nodes.id = nodes_tags.id AND nodes_tags.key='name'
```

```
ORDER BY nodes.version DESC LIMIT 10;
```

```
Бишкек                        | 45
ОсОО "Хостер kg"              | 29
Славянский Восток             | 18
Интернет Магазин XPERT.KG     | 11
Sierra Coffee                 | 11
Скульптурный комплекс "Манас" | 10
Народный (Панфилова)          | 10
Колобок                       | 9
Alex.kg                       | 9
Библиотека №5                 | 9
```

Other ideas about dataset

This section describes additional suggestions for improving and analyzing the data.

Update of street names

After the Collapse of the Sovied Union Kyrgyzstan became an independent country and some of streets with Soviet names was renamed to Kyrgyz names. During auditing I discovered that some data contain old Soviet street names and need to be updated. This website contains information about renaming. The problem is that some old named streets was splitted and different parts got a different new names. So only old street name is not enough to get new street names, we also need to know what part of street is meant in particular node. And retrieving this kind of information seems to be difficult.

Accuracy of postcode

I had a theorethical idea to validate postcodes by street address, using information from official kyrgyz post website. But then I discovered that information on this website is incomplete: some existing postcodes and streets doesn't mentioned there. So This kind of validation is impossible, as far as we don't have complete information about correspondance of postcodes to street address.

Accuracy of website

During auditing websites I discovered that some of websites don't exist. It would be a nice improvement to check programmatically if the websites are reachable. Ablity to check reachability of particular url also would be useful for choosing 'https' scheme when it possible (it's preferred over 'http' scheme). Unfortunately Python < 2.7.9 doesn't include native support for Serever-Name-Indication, which is needed for processing some urls. This makes problematic to use requests library for suggested improvement. However, website.py contains commented implementation, which probably would work on some computers.