Extracting, cleaning and querying the Paris area of Openstreetmap

Introduction:

During this project I have used the paris of OpenStreetMap in XML format. The goal of that project was to clean the informations from that XML file, feed the corrected data into an SQL database and then make interresting requests.

The Paris area interest me a lot because it is the place were i grew up and live. The interest of this area also comes from it size and density of population. The extract used during this project was downloaded the 4th november. I precise the date as it seems that the extracted zone as changed a bit for the mapzen extract since that time. The file size of the extracted XML dataset is 4.5GB.

I will first present the auditing process of the file, then the problem found and correction used. Then explain the interresting fact found by querying the database created.

Data auditing:

This project focused on extracting node and ways informations form the XML file and correcting way names and phone number found in the dataset. The audit of the dataset was done using an audit script (see file: ./Script/1_audit.py), that would iterate on the street names and return the name or only part of it depending on the match of regular expression. After some queries it appears that several problem could be found in this dataset.

Uppercase and accentuation problems:

The first challenge found was that some people would write names only in capital letters while other would write it lowercase. This becomes a problem for French street names because accentuated letters are found in French names and word. Because of the wide distribution of Microsoft Windows and the difficulty to write uppercased accentuated letter on the French keyboard, user would usually write uppercased accentuated letters as the uppercased non-acentutaed letter. For example the word "général" would often be written "GENERAL" instead of "GÉNÉRAL" by a french writer. This becomes a problem because the first idea was to lowercase all names before correcting them and entering them in the database. The step taken to find problematic names was to look for accentuated letters, make a list of all accentuated names and then search them in non accentuated version to make a list of all wrongly accentuated names.

French naming convention and related problems:

French street names have to follow some rules that are not allways in use or used by people. Part of these rules could be found on this wikipedia article, it is in French because I couldn't find a concise but complete article in English. This article explain the usual rules and the discussion around some of these rules. The main rules are:

- Word in the street name should have a capitalised first letter, except for determinant and the type of street ("rue", "boulevard"...).
- Word should be separated by hyphen.

Because I decided to lowercase all street names this will autimatically remove any inconsistencies with the first rule. The second rule is more problematic because this rule is discussed and in a lot of case not used by people. In the majority of cases this rule wasn't applied and to make corrections and queries from the database on the street names easier, I decided to remove hyphens in the rare cases the hyphen rule was used.

Other hyphen problems:

Another use of the hyphen in french is between Peoples names for composed firstnames, for instance in "Jean-Jacques Rousseau", or between familynames for composed family names, for example "Paul Vaillant-Couturier". Another use of hyphen is inside religious names like "Saint-Antoine" or "Notre-Dame". Hyphen are also found in other composed word for example in "avenue d'Alsace-Lorraine". For both use case I kept the hyphen and corrected the names were they were not found.

Apostrophe:

The French language makes an eavy use of the apostrophe, for example in "jeanne d’arc", "de l’amitié". As expressed in this article computer keyboards are not really made to write this character, usually people would use the typewriter apostrophe ' (' ≠ ’). The majority of people would use the typewriter apostrophe, but some contributor have used the normal apostrophe. To standardize writing, I decided to change the regular apostrophe for the typewriter one.

Abreviation:

Some word were abreviated, while abreviation are a good way for human to write faster using full world seem to be a better way to store the information in standardized way. Abreviation like : "dr" for "docteur" or "st" or "st." for "saint" were found. When abreviations were found they were converted to the full word.

Too Long names

Some street names found were really long and involved the name of several street. They seem to come from node where someone would write all informations about the location of the node in one tag. When names corresponding to the identified wrong names, the node was ignored and not imported in the database.

Phone numbers:

Phone numbers in France have 3 parts:

- the country code: +33, can be replaced by 0 if you are calling from within the counrty
- a location code changing depending on your location
- a random 6 digits number

For mobile phones the location code is replaced by 6 or 7 and the random number is 8 digits long. For voIP offers, usually the location code is replaced by 9 and the random number is 8 digits long.

Loocking at the area I cover in this extracion, I should only find phone numbers with a location code starting with 01 (equivalent to +331) or 02 (+332).

When writting a phone number, we can see different trend, first the country code is usually replaced by 0, I decided to replace that with +33 to make these phone number usable by tourists and other non-French people. The other number are usually grouped by 2 separated by space or period. While looking at the phone numbers I found that other standard where used, separation with '-' or even '/' could be found, sometimes even a mix of space and other separation means. I could also find phone numbers with typo like two periods or two spaces... Another problem found was the use of one tag to insert multiple phone

numbers. Auditing phone numbers was done using regular expression, sample of thoses can be found in the ./Script/1_audit.py script. I decided to only correct phone numbers containing 1 phone number. For input not looking like a phone number, I let them pass into the database. To clean phone numbers, I used a regular expression to capture the last 9 digits and then clean it from spacer, add the country code to have a phone number looking like: +33126574982 (this phone number is a fake one).

Data cleaning and database creation:

After having identified all the problem, the chosen way to import corrected data to the database was to create CSV file containing corrected information. Each CSV file would contain the data to fill one table of the database. To clean the data, each streetname found was first checked against a list of bad names (the too long names). Then if the name wasn't listed as a bad name, it was divided between fisrt word and the other part of the name (in French street type is at the start of the name: "***rue*** de l'Europe"). Each part was then checked aginst a dictionnary containing the wrongly written names and its correction. The name was then corrected if necessary and the checked name was the passed to be written into the right CSV. Some extra work checking was done for edge cases. The database was created using the Sqlite API in python following the schema given by Udacity. The import was done directly at the Sqlite shell as it seemed easy for a small database with only a few tables. Because of the way the import function work, I created copies of the CSV files with their header removed because importing CSV into existing table, the file is considered com not having any header by Sqlite3 (see article in the "Existing Table" section). The CSV files created were:

- nodes.csv, 1.5 GB
- nodes_tags.csv, 122 MB
- ways.csv, 185.6 MB
- ways_nodes.csv, 597 MB
- ways_tags.csv, 468.5 MB

The database file is 2.7 GB

Querying the database:

First some basic informations:

1. **number of nodes and ways:**

   The number of nodes and ways is easely found by counting the number of nodeid and way id, here are the queries used:

   ```
   SELECT COUNT(id) FROM nodes;

   SELECT COUNT(id) FROM ways;
   ```

   We found that we have **3128556 ways** and **18957806 nodes**.

2. **number of unique users:**

   Users can be found in the nodes and ways tables. We will query both table to find unique users:

   ```
   SELECT COUNT(DISTINCT(u.users))
   FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) u;
   ```

   I found **7118 unique users**

3. **speed limits in the area:**

   The Paris Area is in majority an urban area. Recently there has been high air polution. Because of the geological situation of Paris and cold air condition, all the dust produced by traffic, heating, industries, has been blocked over Paris. Potitics are more and more trying to reduce the car traffic and want to reduce the speed limits in a lot of area. Within this context I wanted to investigate the repartition of speed limits between the different usual speedlimit.

   Usual speed limits are 30-40-50-60-70-80-90-100-110-120-130 km/h, in some situations limits can be below 30 km/h but this is exceptionnal. Speedlimits can't be over 130 km/h as it is the maximal speedlimit on highways. If we don't limit the query to these speedlimits we find badly formated answer like: FR:30; FR:zone30... For easyest readability only the answer formated in the right way are searched.

```
SELECT tags.value, count(tags.value) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key like '%maxspeed%'
  AND tags.value IN (30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130)
GROUP BY tags.value
ORDER BY count DESC;
```

   we find 32746 references to these speedlimits, in decreasing order of apprearance:

   - 30 km/h: 13871 references ~42%
   - 50 km/h: 12039 references ~37%
   - 90 km/h: 2329 references ~7%
   - 70 km/h: 2166 references ~7%
   - 110 km/h: 1013 references ~3%
   - 120 km/h: 461 references ~1.4%
   - 130 km/h: 345 references ~1%
   - 100 km/h: 275 references ~<1%
   - 40 km/h: 125 references
   - 60 km/h: 103 references
   - 80 km/h: 19 references

   These number doesn't represent the length of street that can be found in each category, but they can be considered an estimator of the proportion of street in each category. The difference between the number of ways (3128556) and the number of speeds limits found (32746 = 1% of the number of way) is quite impressive and reveal that the speed coverage doesn't look complete. A street can have a change in it speed limit in some area and then be counted multiple times. Such speed limit changes are more likely to occur in city than in the coutryside. With this query we have probably overestimated the number of street in city (with limit of 30-50 km/h) but as a first indication we find that 79% of the speed limits are limit in city. Country roads (limit 60-90) account for ~15% of the limit found and highways ~6%. If these estimator represent the reality it is interresting to see the repartition between low capacity street (city street) and high capacity street (country road/highways).

   Despite the repartition of roads between hogh and low capacity, there are still a huge number of traffic jams. The current policy in Paris is to reduce the number of cars in the city and to try to make people use public transport. While this position make sense for Paris one of the problem is that the high capacity roads in the region are organised in a star model (see maps. This organisation is also found for a lot of the public transport for example train. This lead to concentrate the traffic on Paris and reducing the traffic capacity alone in Paris would probably not reduce polution much as people that need to travel from one side to another would accumulate on highways just outside the Paris limit. Regional action to better organize the traffic would be a better solution. The Region is starting to take action, but it would still need some years before the

effect can be seen. The propositions range from new highways to redirect the traffic, reducing the highspeed limits from 130 km/h to 110 km/h according to this rapport.

number of hotels:

The Paris Area being a highly attractive place for tourists, I found interresting to find the number of hotels in the area:

```
SELECT count(distinct(id))
FROM nodes_tags
WHERE value='hotel';
```

We find 1253 nodes corresponding to hotel location in the Paris Area.

Touristic points:

After the hotel it could be to see the interresting points for tourists. I Looked for the 5 more important type of tourism tags excluding hotel from the list:

```
SELECT value, COUNT(id)
FROM nodes_tags
WHERE key='tourism' AND value!='hotel'
GROUP BY value
ORDER BY COUNT(id) DESC
LIMIT 5;
```

It gives us the following results:

```
information  6248
artwork      1065
attraction   293
picnic_site  198
museum       133
```

Train station:

```
SELECT COUNT(DISTINCT(id))
FROM nodes_tags
WHERE (key='railway' or key='public_transport') and value='station';
```

We found 732 train stations.

Imporving possibility:

Phone numbers could be more corrected, the actual regular expression used, doesn't correct if digits are separated with two periods, spaces, if '/' or '(')' are used. The parenthesis wheren't corrected because they weren't found much often and followed different pattern each time which is could not be easily solved programatically. Another improvement could be to create a new tag to separate tag containing two phones numbers into two different tags.

A problem I found while querying the database was when i tried to query the city names. It appears that the same formating problem than the one encoutered on street names can be found. When I run: `SELECT tags.value FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) tags WHERE tags.key='city' GROUP BY value;` we find city with the name spelled in uppercase or hyphen problems... It could be a good idea to standardize the city names.

A problem with standardizing city names is the same as for street names, some user may make request thinking another way of using hyphens or uppercase and they could find wrong results.

Another problem found was in the postcode formating, a lot of strange numbers can be found in the list after running: `SELECT tags.value FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) tags WHERE tags.key='postcode' GROUP BY value;`.

Conclusion:

The Paris Area as already a great coverage but lack some informations like the speedlimit. Also the name formating could be homogenized to remove potential uncertainty and make retrival of data like street names or city names easyer. Some query showed that State or public organisations have imported data into the openstreetmap database. This data is usually imported in a scripted way, some of these tags appear to not be standard and require extra processing to be used. It could be interresting to make an audit of these tags and to find ways to incorporate them better into the database. A project could be to have an automated test every time a user would input data in the OpenStreetmap Database. It doesn't seem realistic to expect that as it would require more server power that would costs, but it could improve the database. For example to make sure that postcode entry are only digits (for France) or to check City names against a cleaned database to check for incorrect spelling... To help with the problem of the low availabitlity of street speed limits, we could imagine an app like Waze that would track actual speed on different street, but also frequentation and give informations to drivers on the frequentation of a street. At the same time it could help find the limitation on that street by analysing the speed of the users. It could even give indications on the difference between speed limit and actual average speed. That sort of app could also be really usefull for the region to analyse users behaviour and identify problemn in the road organisation that the actual car counting system (which only count car at one place, but don't track origin and destination of cars on the roads). can't see.