OPEN STREET MAP DATA WRANGLING WITH SQL

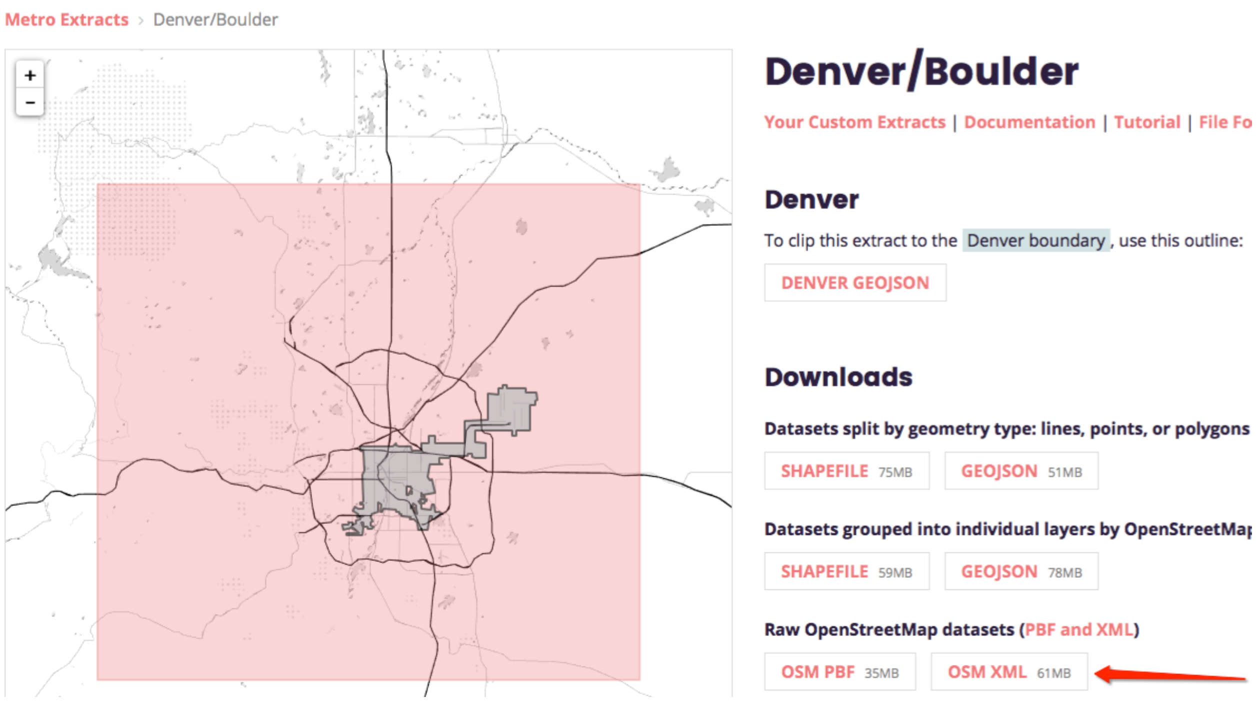
Awad Bin-Jawed

I will demonstrate the process of data wrangling for the city of **Denver, CO, USA**. This project will be an iterative process, so hopefully I will get a desired result as accurate as possible after an extensive cycle of repeating operations. This is why we say this whole cycle is convergent.

I extracted the OSM XML file of Denver.

I will need to audit the street types and standardize the data; this way, I will know exactly what to extract.

Mapzen URL https://mapzen.com/data/metro-extracts/metro/denver- boulder_colorado/85928879/Denver/



Metro Extracts › Denver/Boulder

Denver/Boulder

Your Custom Extracts | Documentation | Tutorial | File Fo

Denver

To clip this extract to the Denver boundary, use this outline:

DENVER GEOJSON

Downloads

Datasets split by geometry type: lines, points, or polygons

SHAPEFILE 75MB | GEOJSON 51MB

Datasets grouped into individual layers by OpenStreetMa

SHAPEFILE 59MB | GEOJSON 78MB

Raw OpenStreetMap datasets (PBF and XML)

OSM PBF 35MB | OSM XML 61MB

DATA AUDITING

The **audit.py** script generates two outputs:

1) **audit.txt**
2) **audit_xsd.txt**

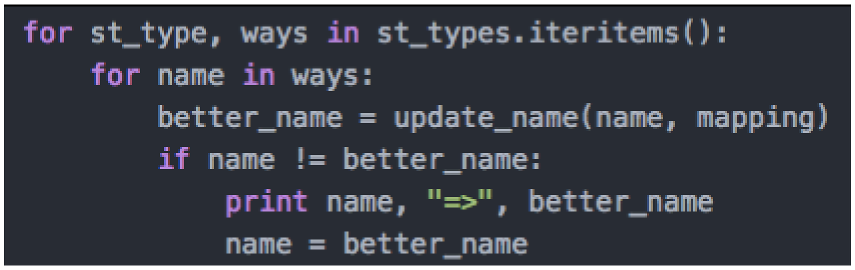
The **audit.txt** contains captured output of 6 functions:

```
print(str(audit_street(OSM_FILE)))

fix_street(audit_street(OSM_FILE))

print(str(audit_city(OSM_FILE)))

print(str(audit_postcode(OSM_FILE)))

output.write(str(audit_phone(OSM_FILE)))

fix_phones(phone_list)
```

The **audit_xsd.txt** contains the result of validation of the full map set.

DATA CLEANING:

The following print statement:



```
for st_type, ways in st_types.iteritems():
    for name in ways:
        better_name = update_name(name, mapping)
        if name != better_name:
            print name, "=>", better_name
            name = better_name
```

This will print the street names after the update of the original, unclean street names.

Essentially, the goal of the scripts is to analyze the data of Denver.

Using the editor (in this case Atom), you may manually analyze the large .osm file.

There is a method run() in all files which can be called from report.py

or in an interpreter.

mapparser.py

This script will yield a defaultdic dictionary of tags and occurrences.

users.py

This script will yield a defaultdic dictionary of user's id numbers.

tagtype.py

This script will yield a simple dictionary of categorized <tag> tags. It also

outputs problematic tags.

DATA OVERVIEW

Step One: Create the .csv files

After auditing, the next step is to prepare the data for insertion to a SQL database. We parse the elements in the OSM XML file, which will transform them from .doc format to tabular format. The process involved using the iterparse through the elements in the XML and shaping the elements into several structures. The schema was to ensure the data is in the correct format. Now we write each data structure to the appropriate .csv files.

Step Two: Upload the .csv files to SQL

Then we want to load the csv files into a database:

Problems Encountered:

sample.py

This script had a broken format:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm>
        <node changeset="7551724" ... />
                <node />
....
                <tag k="type" v="restriction" />
        </relation>
        </osm>
```

This created problems, which were due to output.write() breaking indentations.

The first one can be fixed by output.write('<osm>\n\t'), the second one can be fixed by manually editing the last line in the new file.

data.py

This program creates 4 csv files from denver-boulder_colorado_sample.osm

to be added to the database. The validator in it WILL NOT raise errors because

it doesn't check the nested tags.

The xml+cerberus+schema code is very slow. Approximately, it is MUCH

slower than the lxml+.xsd code.

Anaconda Navigator:

The Anaconda Navigator, one random day, decided to keep crashing, so I decided to get a new, clean install. This resolved the issue of the navigator crashing, but the new install has iPython version 4. The codes in Udacity are catered towards the Python Version 2 environment. I kept getting the "Syntax Error" message.

In order to avoid the Syntax Errors, I resolved this issue by opening the MacOS terminal and installing the python version 2:

```
conda create -n python2 python=2.7 anaconda
```

If I need to ever activate this environment for future purposes, I simply need to enter:

```
source activate python2
```

Additional Ideas

The state of Colorado has recently been grabbing headlines due to a controversial legislative action... the legalization of recreational marijuana in 2012. The Colorado Amendment 64 was a successful measure to overrule the longstanding drug policy of Colorado’s own constitution, which has kept prohibited cannabis since 1917.

It would be interesting to map out cannabis and crime in the state of Colorado. More specifically, the Denver metro area.

The crime CSV file, extracted from the Denver Open Data Catalog comprised of data sets of criminal offenses during the past 5 years (since 2012 until now). See **mapcrime.py** in the repositories

The shops CSV file, extracted from the Open Colorado Data Sets, comprised a listing of active shops/dispensaries as of November 2017. See **mapshops.py** in repositories

Google maps was utilized via pygmaps. I show the first 10 places. You can change the variable count (I use count=10) and get more places on the map.

Some question that may arise:

-How can we be sure if the crimes are drug-related charges? If they are DUI, it would require further inquiry to distinguish between cannabis or other intoxicants (ie, alcohol).

-What about robberies of the shops themselves? Wouldn’t this be considered blaming the victim?

-If someone was engaging in criminal behavior and he had it in his system, was it solely because of cannabis or other factors, such as past criminal/juvenile behavior prior to legalization?

-What if the perpetrator of a respective crime is from outside town, and was visiting Denver?

Some possible consequences:

-Place these dispensaries under severe scrutiny in links are suggested between the shops and acts of crime.

-Create backlash of vilification of cannabis users as dangerous criminals.

-Recriminalizing and resulting prohibition would lead to loss of state revenues from taxing marijuana.

-Undermine proponents of medicinal benefits of cannabis and its funding of research altogether.

Some possible solutions:

-All purchases have documentation of each buyer’s identification. This would link every potential offender to an address of each shop.

-There would be an incumbency of each crime to be followed up by a drug test.

-There would be a necessity of careful and sensitive approach by journalists and reporters to not jump to conclusion and create mass hype or misleading the masses based on hidden agendas of bias reporting or political leanings. If business owners are financially impacted negatively by the media hysteria, they may file class action lawsuits against the reporters and their news corporations for damages (libel).

For a map comparing both crime and shops, See **MapBothCrimeAndShops.py** in repositories.

I show the first 10 places. You can change the variable count (I use count=10) and get more places on the map.

For a map comparing both crime and shops, see **MapBothCrimeAndShops.py** in repositories.