Lesson 6: Case Study – Openstreetmap Data

Iterative Parsing

```
def count_tags(filename):
    tree = ET.parse(filename)
    root = tree.getroot()
    tags = {}
    iter_ = tree.getiterator()
    for elem in iter_:
        if elem.tag in tags:
            tags[elem.tag] += 1
        else:
            tags[elem.tag] = 1
    return tags
```

Tag Types

```
def key_type(element, keys):
    if element.tag == "tag":
        if lower.search(element.attrib['k']):
            keys['lower'] +=1
        elif lower_colon.search(element.attrib['k']):
            keys['lower_colon'] +=1
        elif problemchars.search(element.attrib['k']):
            keys['problemchars'] +=1
        else:
            keys['other'] +=1
    return keys
```

Exploring Users

```
def process_map(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if element.tag == "node" or element.tag == "way" or element.tag == "relation":
            if element.attrib['user'] not in users:
                users.add(element.attrib['user'])
    return users
```

Improving Street Names

```
mapping = { "St": "Street", "St.": "Street", "Ave":"Avenue", "Rd.":"Road"}
def update_name(name, mapping):
    fix = name.split()[-1]
    name = name.replace(fix, mapping[fix])
    return name
```

Preparing for Database

```
def shape_element(element):
    node = {}
    if element.tag == "node" or element.tag == "way" :
        node['id'], node['type'] = element.attrib['id'], element.tag
        try:
            node['visible'] =  element.attrib['visible']
        except KeyError:
            node['visible'] = 'false'
        node['created'] = {"version":element.attrib['version'], "changeset":element.attrib['changeset'],
"timestamp":element.attrib['timestamp'], "user":element.attrib['user'], "uid":element.attrib['uid']}
        try:
            node['pos'] = [float(element.attrib['lat']), float(element.attrib['lon'])]
        except KeyError:
            pass
        address = {}
        for tag in element.iter("tag"):
            if problemchars.search(tag.attrib['k']):
                pass
            elif tag.attrib['k'][0:5] == 'addr:':
                if tag.attrib['k'].count(':') <= 1:
                    if 'housenumber' in tag.attrib['k']:
                        address['housenumber'] = tag.attrib['v']
                    if 'postcode' in tag.attrib['k']:
                        address['postcode'] = tag.attrib['v']
                    if 'street' in tag.attrib['k']:
                        address['street'] = tag.attrib['v']
            else:
                node[tag.attrib['k']] = tag.attrib['v']
        if address:
            node['address'] = address

        nodes = []
        for nodess in element.iter("nd"):
            nodes.append(nodess.attrib['ref'])
        if nodes:
            node['node_refs'] = nodes
        return node
    else:
        return None
```

Final Project Code

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Code to parse Openstreetmaps (osm) data, to analyse, clean and convert to json for
uploading to mongodb for further analysis
"""

import xml.etree.cElementTree as ET
from collections import defaultdict
from pymongo import MongoClient
import re
import pprint
import codecs
import json
import pprint

osm_file = open('sydney_australia.osm', 'r')

street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
street_types = defaultdict(set)

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

expected = ['Street', 'Avenue', 'Boulevard', 'Drive', 'Road', 'Court', 'Place',
'Circuit', 'Lane', 'Parade', 'Crescent', 'Highway', 'Way', 'Close']

mapping = { "St": "Street", "St.": "Street","st": "Street", "street": "Street",
"Ave":"Avenue", "Av.":"Avenue", "Rd":"Road", "Rd.":"Road", "road":"Road",
"Hwy":"Highway", "place":"Place"}

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)

def print_sorted_dict(d):
    keys = d.keys()
    keys = sorted(keys, key=lambda s: s.lower())
    for k in keys:
        v = d[k]
        print "%s:%d" % (k, v)

def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

def audit():
    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 is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    pprint.pprint(dict(street_types))

def update_name(name, mapping):
    fix = name.split()[-1]
    try:
        name = name.replace(fix, mapping[fix])
    except KeyError:
        pass
    return name

def better_name():
    for st_types, ways in street_types.iteritems():
        for name in ways:
            better_name = update_name(name, mapping)
            if better_name == name:
                print 'not fixed: ', name
            else:
                print name, "=>", better_name

def count_tags(filename):
    tree = ET.parse(filename)
    root = tree.getroot()
    tags = {}
    iter_ = tree.getiterator()
    for elem in iter_:
        if elem.tag in tags:
            tags[elem.tag] += 1
        else:
            tags[elem.tag] = 1
    pprint.pprint(tags)

def process_map(file_in, pretty = False):
    # You do not need to change this file
    file_out = "{0}.json".format(file_in)
    data = []
    with codecs.open(file_out, "w") as fo:
        for _, element in ET.iterparse(file_in):
            el = shape_element(element)
            if el:
                data.append(el)
                if pretty:
                    fo.write(json.dumps(el, indent=2)+"\n")
                else:
                    fo.write(json.dumps(el) + "\n")
    return

def shape_element(element):
    node = {}
    if element.tag == "node" or element.tag == "way" :
        node['id'], node['type'] = element.attrib['id'], element.tag
        try:
            node['visible'] =  element.attrib['visible']
        except KeyError:
            node['visible'] = 'false'
        node['created'] = {"version":element.attrib['version'],
"changeset":element.attrib['changeset'], "timestamp":element.attrib['timestamp'],
"user":element.attrib['user'], "uid":element.attrib['uid']}
        try:
            node['pos'] = [float(element.attrib['lat']),
float(element.attrib['lon'])]
        except KeyError:
            pass
        address = {}
        for tag in element.iter("tag"):
            if problemchars.search(tag.attrib['k']):
                pass
            elif tag.attrib['k'][0:5] == 'addr:':
                if tag.attrib['k'].count(':') <= 1:
                    if 'housenumber' in tag.attrib['k']:
                        address['housenumber'] = tag.attrib['v']
                    if 'postcode' in tag.attrib['k']:
                        address['postcode'] = tag.attrib['v']
                    if 'street' in tag.attrib['k']:

                        better_name = update_name(tag.attrib['v'], mapping)
                        if better_name == tag.attrib['v']:
                            pass
                        else:
                            print tag.attrib['v'], "=>", better_name

                        address['street'] = tag.attrib['v']
            else:
                node[tag.attrib['k']] = tag.attrib['v']

        if address:
            node['address'] = address

        nodes = []
        for nodess in element.iter("nd"):
            nodes.append(nodess.attrib['ref'])
        if nodes:
            node['node_refs'] = nodes
        return node
    else:
        return None

def add_data(db):
    with open('sydney_australia.osm.json', 'r') as f:
        for line in f:
            db.openmaps.insert(json.loads(line))
    f.close

def query_db(db):
    total = db.openmaps.find({'type':'node'}).count()
    print 'Total nodes = ', total
    total = db.openmaps.find({'type':'way'}).count()
    print 'Total ways = ', total
    total = db.openmaps.find().count()
    print 'Total nodes and ways = ', total

    pipeline = [{'$group': {'_id':'$created.user', 'count':{'$sum':1}}}, {'$sort':
{'count':-1}}, {"$limit":5}]
    result = db.openmaps.aggregate(pipeline)
    print 'Prolific users:'
    for i in result:
        pprint.pprint(i)

    pipeline = [{'$group': {'_id':'$created.user', 'count':{'$sum':1}}}, {'$project':
{'_id':'$_id', 'percent':{'$divide':['$count', total]}}}, {'$sort': {'percent':-1}},
{"$limit":5}]
    result = db.openmaps.aggregate(pipeline)
    print '% edits:'
    for i in result:
        pprint.pprint(i)

    pipeline = [{'$match':{'address.postcode':{'$exists':1}}},
{'$group':{'_id':'$address.postcode', 'count':{'$sum':1}}}, {'$sort': {'_id':-1}},
{'$limit':5}]
    result = db.openmaps.aggregate(pipeline)
    print 'top postcodes:'
    for i in result:
        pprint.pprint(i)


    print len(db.openmaps.distinct('amenity'))
    print len(db.openmaps.distinct('created.user'))
    pipeline = [{'$match':{'amenity':{'$exists':1}}}]

    pipeline = [{'$match':{'amenity':{'$exists':1}}}, {'$group':{'_id':'$amenity',
'count':{'$sum':1}}}, {'$sort': {'count':1}}, {'$match':{'count':1}}]
    pipeline = [{'$match':{'amenity':{'$exists':1}}}, {'$group':{'_id':'$amenity',
'count':{'$sum':1}}}, {'$group':{'_id':'total amenity', 'count':{'$sum':'$count'}}}]
    result = db.openmaps.aggregate(pipeline)
    for i in result:
        pprint.pprint(i)

    pipeline = [{'$match':{'amenity':{'$exists':1}}}, {'$group':{'_id':'$amenity',
'count':{'$sum':1}}}, {'$sort': {'count':-1}}, {'$limit':10}]
    result = db.openmaps.aggregate(pipeline)
    print 'top amenities:'
    for i in result:
        pprint.pprint(i)

    pipeline = [{'$match':{'amenity':{'$exists':1}, 'amenity':'school'}},
{'$group':{'_id':'$address.postcode', 'count':{'$sum':1}}}, {'$sort': {'count':-1}},
{'$limit':10}]
    result = db.openmaps.aggregate(pipeline)
    print 'pubs:'
    for i in result:
        pprint.pprint(i)

def get_db():
    from pymongo import MongoClient
    client = MongoClient('localhost:27017')
    db = client['openmaps']
    return db

if __name__ == "__main__":
    count_tags('sydney_australia.osm')
    audit()
    better_name()
    process_map('sydney_australia.osm', False)
    db = get_db()
    add_data(db)
    query_db(db)
```