import xml.etree.ElementTree as ET

from collections import defaultdict

import pprint

import re

from pymongo import MongoClient

lower = re.compile(r'^([a-z]|\_)\*$')

lower\_colon = re.compile(r'^([a-z]|\_)\*:([a-z]|\_)\*$')

problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

street\_type\_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

CREATED = [ "version", "changeset", "timestamp", "user", "uid"]

expected = ["street", "avenue", "boulevard", "drive", "court", "place", "square", "lane", "road",

"trail", "parkway", "commons", "close", "crescent", "way"]

# UPDATE THIS VARIABLE

mapping = { "st": "Street",

"st.": "Street",

"ave": "Avenue",

"rd.": "Road",

"rd": "Road",

"roads": "road",

"cresent": "crescent"

}

def test():

filename = "johannesburg\_south-africa.osm"

#Get all tags + count of top level tags

topTagTypes = count\_tags(filename)

#Get a list of all sub tags classified as either (lowers, lower\_colon, problemchar & other) + counter

tagTypeBreakdown = process\_map(filename)

#Get a list of all the street types

endingStreetNameTypes = process\_streetEnding(filename)

dataToUpload = prepForUpload(filename)

pprint.pprint(dataToUpload)

uploadDataToDB(dataToUpload)

#What tags do we have?

def count\_tags(filename):

dictTags = {}

for event, elem in ET.iterparse(filename):

if elem.tag in dictTags:

dictTags[elem.tag] +=1

else:

dictTags[elem.tag] = 1

return dictTags

#What type (lowers, lower\_colon, problemchar & other) of nested keys do we have?

def process\_map(filename):

keys = {"lower": 0, "lower\_colon": 0, "problemchars": 0, "other": 0}

for \_, element in ET.iterparse(filename,events=("start",)):

keys = key\_type(element, keys)

return keys

#Part of process map

def key\_type(element, keys):

if element.tag == "tag":

if re.match(lower, element.attrib['k']):

keys['lower'] +=1

elif re.match(lower\_colon, element.attrib['k']):

keys['lower\_colon'] +=1

elif re.match(problemchars, element.attrib['k']):

keys['problemchars'] +=1

else:

keys['other'] +=1

return keys

#Process Node & Way street name endings

def process\_streetEnding(filename):

street\_types = defaultdict(set)

for event, elem in ET.iterparse(filename, 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'])

return street\_types

#Check if tag is a street name

def is\_street\_name(elem):

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

#Add street name endings to list of street\_type

def audit\_street\_type(street\_types, street\_name):

m = street\_type\_re.search(street\_name)

if m:

street\_type = m.group().lower()

if street\_type not in expected:

street\_types[street\_type].add(street\_name.lower())

#Change street name ending

def update\_name(name):

for item in mapping:

if name.find(item) > 0:

name = name[:name.find(item)] + " " + mapping[item]

return name

return name

#Creating elements for importing to Mongo

def shape\_element(element):

node = {}

address = {}

if element.tag == "node" or element.tag == "way" :

#Create created tag dictionary

createdTag = {}

for tag in CREATED:

createdTag[tag] = element.attrib[tag]

node['created'] = createdTag

#Getting co-ordinates

pos= [0,0]

if ('lat' in element.attrib) and ('lon' in element.attrib):

pos[0] = float(element.attrib['lat'])

pos[1] = float(element.attrib['lon'])

node['pos'] = pos

#Getting through ktags

for tag in element.iter("tag"):

if (re.match(problemchars,tag.attrib['k'])) or (tag.attrib['k'].count(':') >1):

pass

else:

if (tag.attrib['k'].startswith("addr:")):

if tag.attrib['k'] == "addr:street":

address['street'] = update\_name(tag.attrib['v'])

else:

address[tag.attrib['k'].split(':')[1]] = tag.attrib['v']

elif tag.attrib['k'].count(':') == 1:

node[tag.attrib['k'].split(':')[1]] = tag.attrib['v']

else:

node[tag.attrib['k']] = tag.attrib['v']

#Adding the other tag

node['address'] = address

node['id'] = element.attrib['id']

node['type'] = element.tag

return node

else:

return

#Prepare data for upload

def prepForUpload(filename):

data = []

for event, elem in ET.iterparse(filename):

el = shape\_element(elem)

if el:

data.append(el)

return data

def uploadDataToDB(nodesToImport):

client = MongoClient('localhost', 27017)

db = client.project2Submitdb

streetNodes = db.streetdata

for node in nodesToImport:

streetNodes.insert(node)

if \_\_name\_\_ == "\_\_main\_\_":

test()