Clean and Convert MapData

June 22, 2015

```
In [1]: import xml.etree.cElementTree as ET
        import pprint
        import re
        import codecs
        import json

        OSMFILE = "FernCreek-Highview_Kentucky.osm"  # The map data...
        street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)  # Regular Expression to pick off the

        # An audit of the data revealed 3 types to correct: Road and Point
        mapping = { "Rd": "Road",
                    "PT": "Point",
                    }

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

        # keys requiring special processing...
        CREATED = [ "version", "changeset", "timestamp", "user", "uid"]


        '''
        Pick off the street type and map it against a known list.
        '''
        def update_name(name):
            m = street_type_re.search(name)
            if m:
                street_type = m.group()
                if mapping.has_key(street_type):
                    name = re.sub(street_type_re, mapping[street_type], name)
            return name


        '''
        Shape the element into the proper format, fixing the problem fields along the way...
        '''
        def shape_element(element):
            worship = False
            fuel = False
            node = {}
            if element.tag == "node" or element.tag == "way" :
                node['type'] = element.tag  # Start with setting 'type' to 'node' or 'way'
```

```
# Now, process the attributes in the node/way tag
for k in element.attrib:
    # If the attribute is one that belongs in a subdocument called 'created',
    # then add the 'created' section (if not already there), and then insert
    # the attribute and value
    if k in CREATED:
        if not node.has_key('created'):
            node['created'] = {}
        node['created'][k] = element.attrib[k]
        continue

    # If the attribute is longitude or latitude, then we create a 'pos' (position)
    # array that will be used in a geospatial index.  This actually runs twice, one
    # for lon and again for lat.  It causes no issues to update the field the second ti
    # and makes the code cleaner than checking if it already exists.
    if k == 'lon' or k == 'lat':
        node['pos'] = [float(element.attrib['lat']),float(element.attrib['lon'])]
        continue

    node[k] = element.attrib[k]  # No special processing required, so just add it.

# Now, we process the subtags under node or way...
for tag in element.iter("tag"):
    # Check for problem characters
    if not re.search(problemchars, tag.attrib['k']):
        # Split on : to pull off the first part of the attribute name
        ks = tag.attrib['k'].split(':')
        # Now, check to see if it's an address attribute.  If yes, do further checking
        # adding it to the 'address' subdocument; otherwise, just add it.
        if ks[0] == "addr":
            # If split returned 3 or more fields, this means there was a second colon i
            # attribute name, so we skip it.  If not, we add it to the 'address' docume
            if len(ks) < 3:
                if not node.has_key('address'):
                    node['address'] = {}
                if ks[1] == 'street':   # If street, then fix the name; otehrwise add t
                    street_name = update_name(tag.attrib['v'])
                    node['address'][ks[1]] = street_name
                else:
                    node['address'][ks[1]] = tag.attrib['v']
        else:
            node[tag.attrib['k']] = tag.attrib['v']  # always add the pair
            '''
            If the amenity is a place of worship or fuel station, then update/set the b
            '''
            if tag.attrib.get('k') == "amenity" and tag.attrib.get('v') == "place_of_wo
                node['building'] = 'yes'
                worship = True
            if tag.attrib.get('k') == "amenity" and tag.attrib.get('v') == "fuel":
                node['building'] = 'yes'
                fuel = True
            if tag.attrib.get('k') == "building":
                if worship or fuel:
                    node['building'] = 'yes'
```

```
                            else:
                                node['building'] = tag.attrib['v']


                # Now that we've processed all the subtags, we need one last processing special for way
                # We need to find all teh node 'nd' tags and add them to an array called node_refs.
                if element.tag == "way":
                    for tag in element.iter("nd"):
                        if not node.has_key('node_refs'):
                            node['node_refs'] = []
                        node['node_refs'].append(tag.attrib['ref'])
                return node
            else:
                return None

        '''
        Main Code:
        - open the map file
        - initialize out counters and sets for storing what we find in the data
        - process the data

        Wrangle the data and transform the shape of the data into the model used in Problem Set 6.
        In particular the following things are done:
        - you should process only 2 types of top level tags: "node" and "way"
        - all attributes of "node" and "way" should be turned into regular key/value pairs, except:
            - attributes in the CREATED array should be added under a key "created"
            - attributes for latitude and longitude should be added to a "pos" array,
              for use in geospacial indexing. Make sure the values inside "pos" array are floats
              and not strings.
        - if second level tag "k" value contains problematic characters, it should be ignored
        - if second level tag "k" value starts with "addr:", it should be added to a dictionary "addres
        - if second level tag "k" value does not start with "addr:", but contains ":", you can process
          same as any other tag.
        - if there is a second ":" that separates the type/direction of a street,
          the tag should be ignored
        - for "way" specifically, convert <nd> into a node_refs array
        '''

        osm_file = open(OSMFILE, "r")
        file_out = "{0}.json".format(OSMFILE)
        data = []
        print "Processing map data..."
        with codecs.open(file_out, "w") as fo:
            for event, element in ET.iterparse(osm_file):
                el = shape_element(element)
                if el:
                    data.append(el)
                    fo.write(json.dumps(el, indent=2)+"\n")
        print "Finished processing."
```

```
Processing map data...
Finished processing.
```

In [ ]: