OpenStreetMap数据探索

1.节点检查

In [1]:
```
import xml.etree.ElementTree as ET
import pprint
import re

OSM_FILE_NAME = "map.xml"
```

In [2]:
```
NODE_TYPES = set()

def get_node_types(file_name="sample.xml"):
    context = ET.iterparse(file_name, events=('start', 'end'))
    _, root = next(context)
    for event, elem in context:
        if event=='start':
            #print('start----'+elem.tag)
            NODE_TYPES.add(elem.tag)
```

In [3]:
```
get_node_types(OSM_FILE_NAME)
print(NODE_TYPES)
```

```
{'nd', 'remark', 'tag', 'way', 'member', 'bounds', 'relation', 'node', 'note', 'meta'}
```

In [2]:
```
CHECK_NODES = ["relation", "node", "way"]

way_attribs=set()
way_tag_attribs=set()
node_attribs=set()
node_tag_attribs=set()
relation_attribs=set()
relation_tag_attribs=set()
relation_member_attribs=set()

def get_basic_attributes(file_name="sample.xml"):
    context = ET.iterparse(file_name, events=('start',))
    _, root = next(context)
    for event, elem in context:
        if event=='start' and elem.tag=='node':
            for attrib in elem.attrib.keys():
                node_attribs.add(attrib)
            for node_tag in elem.iter("tag"):
                for attrib in node_tag.attrib.keys():
                    node_tag_attribs.add(attrib)
        elif event=='start' and elem.tag=='way':
            for attrib in elem.attrib.keys():
                way_attribs.add(attrib)
            for way_tag in elem.iter("tag"):
                for attrib in way_tag.attrib.keys():
                    way_tag_attribs.add(attrib)
        elif event=='start' and elem.tag=='relation':
            for attrib in elem.attrib.keys():
                relation_attribs.add(attrib)
            for relation_tag in elem.iter("tag"):
                for attrib in relation_tag.attrib.keys():
                    relation_tag_attribs.add(attrib)
            for relation_member in elem.iter("member"):
                for attrib in relation_member.attrib.keys():
                    relation_member_attribs.add(attrib)
```

In [3]:
```
get_basic_attributes("map.xml")
```

In [6]:
```
#输出各类标签的 attribute 类型
print("way_attribs:=" + str(way_attribs))
print("way_tag_attribs:="+ str(way_tag_attribs))
print("node_attribs:="+ str(node_attribs))
print("node_tag_attribs:="+ str(node_tag_attribs))
print("relation_attribs:="+ str(relation_attribs))
print("relation_tag_attribs:="+ str(relation_tag_attribs))
print("relation_member_attribs:="+ str(relation_member_attribs))
```

```
way_attribs:={'version', 'timestamp', 'id', 'changeset', 'uid', 'user'}
way_tag_attribs:={'k', 'v'}
node_attribs:={'version', 'lat', 'timestamp', 'id', 'changeset', 'uid', 'lon', 'user'}
node_tag_attribs:={'k', 'v'}
relation_attribs:={'version', 'timestamp', 'id', 'changeset', 'uid', 'user'}
relation_tag_attribs:={'k', 'v'}
relation_member_attribs:={'ref', 'type', 'role'}
```

检查node,way,relation中的TAG标签的K值

In [19]:
```
WAY_TAG_K=set()
NODE_TAG_K=set()
REL_TAG_K=set()

def get_tag_k(file_name="sample.xml"):
    context = ET.iterparse(file_name, events=('start',))
    _, root = next(context)
    for event, elem in context:
        if event=='start' and elem.tag=='way':
            for way_tag in elem.iter("tag"):
                WAY_TAG_K.add(way_tag.attrib['k'])

        elif event=='start' and elem.tag=='node':
            for node_tag in elem.iter("tag"):
                NODE_TAG_K.add(node_tag.attrib['k'])

        elif event=='start' and elem.tag=='relation':
            for rel_tag in elem.iter("tag"):
                REL_TAG_K.add(rel_tag.attrib['k'])
```

In [21]:
```
get_tag_k("map.xml")
```

检查每一个K值出现的次数

In [28]:
```
way_tag_k_counter = {k_name:0 for k_name in WAY_TAG_K}
node_tag_k_counter = {k_name:0 for k_name in NODE_TAG_K}
rel_tag_k_counter = {k_name:0 for k_name in REL_TAG_K}
```

In [30]:
```
def count_K_field(file_name="sample.xml"):
    context = ET.iterparse(file_name, events=('start',))
    _, root = next(context)
    for event, elem in context:
        if event=='start' and elem.tag=='way':
            for way_tag in elem.iter("tag"):
                way_tag_k_counter[way_tag.attrib['k']] +=1

        elif event=='start' and elem.tag=='node':
            for node_tag in elem.iter("tag"):
                node_tag_k_counter[node_tag.attrib['k']] +=1

        elif event=='start' and elem.tag=='relation':
            for rel_tag in elem.iter("tag"):
                rel_tag_k_counter[rel_tag.attrib['k']] +=1
```

In [31]:
```
count_K_field("map.xml")
```

In [34]:
```
print(node_tag_k_counter, way_tag_k_counter, rel_tag_k_counter)
```

{'name:lt': 2, 'stars': 6, 'sport': 34, 'tower:type': 18, 'fuel:octane_98': 2, 'seats': 9, 'is_in:country_code': 7, 'toilets:wheelchair': 1, 'highway': 5133, 'diplomatic': 1, 'service:bicycle:repair': 2, 'artist:wikidata': 2, 'artist_name': 5, 'gns:UNI': 12, 'layer': 50, 'wikipedia': 17, 'cuisine': 426, 'place': 222, 'seamark:light:character': 2, 'heritage': 36, 'barrier': 980, 'addr:floor': 2, 'seamark:mooring:category': 1, 'source': 6716, 'gns:DSG': 13, 'ele': 7, 'fee': 51, 'junction': 1, 'is_in:iso_3166_2': 1, 'smoking': 41, 'food': 4, 'internet_access:ssid': 1, 'wifi': 1, 'motorcycle': 13, 'surveillance': 2, 'exit_to:en': 9, 'entrance': 77, 'name:fa': 1, 'healthcare': 2, 'drink': 2, 'ref:vi': 1, 'railway': 1811, 'rooms': 1, 'fuel:diesel': 2, 'height': 2714, 'organic': 2, 'vehicle': 1, 'wpt_symbol': 3, 'notes': 3, 'network': 223, 'crossing_ref': 82, 'name:hr': 1, 'artwork_type': 28, 'trolleybus': 46, 'local_ref': 21, 'floor': 2, 'surface': 1, 'vending': 2, 'source:ref': 10, 'addr:flats': 1, 'description': 50, 'subway': 419, 'name:he': 4, 'parking': 36, 'seamark:information': 1, 'brand': 10, 'toilets:disposal': 37, 'generator:source': 50, 'car': 5, 'price': 12, 'drive_through': 4, 'hiking': 1, 'proposed:railway': 1, 'amenity': 3408, 'spaces': 1, 'memorial:type': 1, 'fuel:octane_92': 3, 'theatre': 2, 'access': 122, 'heritage:ref': 17, 'power': 11710, 'note:plack': 1, 'religion': 8, 'name:jbo': 1, 'bench': 33, 'denomination': 2, 'artwork': 2, 'traffic_signals': 178, 'climbing:boulder': 2, 'ref': 1795, 'website': 230, 'china_class': 6, 'built': 4, 'traffic_signals:sound': 25, 'name:zh_pinyin': 17, 'dsscreated': 74, 'addr:housenumber': 1348, 'truck': 1, 'name:sk': 2, 'artist_name:en': 4, 'opening_hours': 247, 'authentication:membership_card': 1, 'name:zh-hant-CN': 8, 'tree': 1, 'aeroway': 570, 'wikipedia:zh': 3, 'piste:type': 1, 'name:pl': 3, 'name:ro': 1, 'backrest': 17, 'supervised': 2, 'seamark:beacon_special_purpose:colour': 1, 'seamark:light:height': 2, 'ref:cn': 18, 'seamark:light:range': 2, 'target': 1, 'wikidata': 32, 'takeaway': 21, 'recycling:glass': 1, 'capacity': 27, 'name:tr': 1, 'hoops': 1, 'government': 2, 'construction': 2, 'bin': 1, 'building:levels': 37, 'addr:place': 5, 'playce_number': 1, 'name:ia': 1, 'entrance:main': 2, 'contact:phone': 7, 'name:ja': 16, 'designation': 11, 'waterway:sign': 1, 'lanes': 1, 'population:date': 1, 'visibility': 1, 'collection_times': 3, 'payment:bitcoin': 4, 'tourism': 544, 'fire_hydrant:type': 16, 'name:el': 3, 'name:zh-CN': 8, 'display': 1, 'traffic_calming': 128, 'destroyed': 1, 'name:af': 1, 'advertising': 15, 'male': 2, 'FIXME': 1, 'addr:street': 1615, 'name:ku': 1, 'seamark:light:colour': 2, 'seamark:name': 2, 'foot': 114, 'gns:UFI': 13, 'note': 89, 'name:sv': 1, 'internet_access:fee': 15, 'surveillance:zone': 1, 'covered': 5, 'name:zh-hant': 4, 'level': 17, 'name:pt': 12, 'shop': 1257, 'is_in:continent': 7, 'fixme': 17, 'disused': 1, 'name:ru': 10, 'name:cs': 3, 'office': 124, 'payment:litecoin': 1, 'shelter': 42, 'website:en': 87, 'addr:province': 18, 'indoor': 8, 'currency:CNY': 5, 'name:th': 1, 'internet_access': 125, 'tactile_paving': 2, 'booking': 13, 'smoothness': 1, 'clothes': 5, 'ferry': 1, 'is_in:province': 6, 'pet': 1, 'dss created': 1, 'population': 7, 'url': 4, 'outdoor_seating': 17, 'name:eo': 2, 'information': 5, 'branch': 1, 'microbrewery': 2, 'country': 9, 'addr:postcode': 386, 'note:de': 1, 'addr:street:alley': 1, 'unisex': 3, 'bus': 177, 'function': 1, 'train': 5, 'phone': 159, 'traffic_signals:vibration': 25, 'contact:email': 1, 'crossing': 216, 'min_height': 1, 'operator:en': 1, 'addr:country': 28, 'operator': 282, 'service:bicycle:chain_tool': 1, 'NOTE': 1, 'drive_in': 4, 'commemorates:wikidata': 1, 'public_transport': 960, 'seamark:light:reference': 2, 'name:ko': 35, 'service:bicycle:retail': 2, 'historic': 43, 'seamark:light:group': 1, 'addr:street:en': 30, 'name:fr': 15, 'name:ca': 1, 'voltage-high': 2, 'horse': 20, 'landuse': 10, 'building': 169, 'bicycle': 119, 'female': 2, 'short_name': 8, 'fastpass': 3, 'leisure': 115, 'seamark:type': 3, 'name:cn': 1, 'healthcare:speciality': 1, 'craft': 6, 'heritage:operator': 35, 'name:de': 42, 'motorcycle:type': 1, 'addr:neighbourhood': 4, 'generator:method': 5, 'natural': 3948, 'name:it': 3, 'is_in:country': 7, 'service': 1, 'addr:city': 860, 'man_made': 120, 'fuel:octane_95': 3, 'direction': 1, 'name:sr': 1, 'support': 15, 'admin_level': 8, 'scooter': 3, 'inscription': 1, 'map_type': 1, 'motor_vehicle': 33, 'start_date': 22, 'emergency': 48, 'addr:housename': 42, 'unpatrolled': 1, 'diet:halal': 1, 'name:es': 15, 'diaper': 2, 'gns:ADM1': 13, 'addr:district': 21, 'source:name': 20, 'number': 1, 'addr:full': 2, 'fax': 1, 'lit': 15, 'waterway': 154, 'nat_name': 1, 'alt_name:en': 7, 'inscription_1': 1, 'wikipedia:en': 3, 'leaf_type': 2629, 'email': 19, 'name:vi': 19, 'motorcar': 11, 'atm': 100, 'frequency': 6, 'addr:door': 4, 'name:ar': 1, 'recycling:cans': 5, 'name:uk': 1, 'material': 12, 'surveillance:type': 3, 'electricalboat_rental': 1, 'seamark:light:period': 2, 'recycling:paper': 5, 'exit_to': 13, 'weather_protection': 13, 'cinema:3D': 1, 'name:en': 3010, 'colour': 9, 'alt_name:vi': 1, 'addr:unit': 1, 'int_name': 27, 'old_ref': 1, 'rank': 1, 'name': 6215, 'name:hu': 1, 'addr:room': 1, 'artist_name:zh_pinyin': 4, 'leaf_cycle': 2635, 'route_ref': 30, 'delivery': 7, 'alt_name': 14, 'source:population': 1, 'name:kn': 1, 'golf': 37, 'note:en': 1, 'name:da': 1, 'old_name': 15, 'created_by': 2272, 'noexit': 32, 'wheelchair': 31, 'name:zh': 1256, 'name:bg': 1, 'name:io': 1, 'addr:state': 5, 'website_1': 1, 'ref:en': 1, 'memorial': 8, 'service:bicycle:pump': 3, 'billiards:pool': 1} {'roof:material': 230, 'toilets:wheelchair': 1, 'building:level': 3, 'roof:angle': 86, 'tower': 1, 'turn:lanes': 50, 'barrier': 1251, 'honk': 1, 'park_ride': 4, 'bicycle_road': 3, 'fee': 64, 'incline': 47, 'junction': 69, 'ele': 262, 'ramp:wheelchair': 4, 'healthcare': 14, 'destination': 21, 'railway': 2766, 'rooms': 9, 'height': 2654, 'artwork_type': 1, 'crop': 1, 'surface': 1606, 'tiger:PLACEFP': 1, 'parking': 168, 'history': 2, 'amenity': 1770, 'trail_visibility': 8, 'fuel:octane_92': 1, 'access': 2308, 'bridge': 11882, 'parking:condition:right:time_interval': 1, 'power': 376, 'military': 2, 'fence_type': 46, 'bench': 36, 'ref': 5059, 'website': 103, 'destination:ref:to': 2, 'addr:housenumber': 3027, 'aeroway': 2231, 'circuits': 24, 'supervised': 23, 'wikidata': 100, 'building:levels': 3907, 'addr:place': 1, 'destination:street': 66, 'mtb:scale': 6, 'use': 2, 'school': 1, 'int_ref': 360, 'alt_name:zh_pinyin': 21, 'addr:street': 2437, 'foot': 1948, 'covered': 386, 'name:pt': 3, 'boundary': 344, 'website:en': 40, 'ford': 2, 'seamark:cable_submarine:category': 1, 'addr:province': 41, 'destination:lanes': 19, 'social_facility': 6, 'oot': 1, 'oneway:electric_scooter': 1, 'tiger:county': 1, 'source_1': 1, 'bridge:structure': 1, 'monorail': 2, 'landuse': 5243, 'healthcare:speciality': 2, 'name:cn': 3, 'length': 4, 'heritage:operator': 396, 'addr:city_1': 1, 'natural': 1288, 'Commercial': 1, 'is_in:country': 1, 'man_made': 395, 'code': 2, 'navigable': 1, 'motor_vehicle': 455, 'addr:housename': 627, 'emergency': 24, 'name:es': 6, 'building:colour': 1636, 'aerodrome:type': 1, 'lit': 365, 'check_date': 1, 'email': 5, 'motorcar': 84, 'atm': 10, 'frequency': 1186, 'cycleway': 2100, 'tunnel': 673, 'name:uk': 4, 'communication': 1, 'building_1': 1, 'name': 29313, 'tat_deeeee': 1, 'name:no': 1, 'created_by': 15, 'elevator': 1, 'website:zh': 1, 'diplomatic': 1, 'note:height': 64, 'highway': 60837, 'cuisine': 22, 'heritage': 422, 'width': 320, 'wetland': 4, 'lock': 4, 'official_name': 31, 'cycleway:left': 15, 'roof:colour': 2188, 'description': 30, 'lanes:forward': 62, 'subway': 21, 'name:he': 1, 'roundabout': 1, 'boat': 24, 'electrified': 1529, 'denomination': 13, 'contact:twitter': 2, 'relation': 1, 'dsscreated': 2, 'wikipedia:zh': 3, 'old_name:fr': 2, 'attraction': 6, 'ref:cn': 4, 'capacity': 26, 'step_count': 7, 'construction': 281, 'name:ja': 5, 'line': 15, 'male': 1, 'kerb': 2, 'name:ang': 1, 'oneway': 20171, 'destination:to': 19, 'name:zh-hant': 919, 'level': 66, 'image': 7, 'shop_3': 1, 'internet_access': 32, 'cycleway:right': 875, 'noname': 2, 'operational_status': 4, 'country': 4, 'bus': 31, 'contents': 1, 'maxspeed': 873, 'historic': 27, 'fastpass': 8, 'route': 30, 'name:de': 67, 'addr:city': 772, 'admin_level': 270, 'electric_scooter': 7, 'duration': 1, 'conveying': 7, 'addr:district': 52, 'source:name': 4, 'shop_1': 2, 'fax': 4, 'leaf_type': 65, 'gauge': 1683, 'material': 1, 'int_name': 3636, 'roof:orientation': 280, 'short_name:en': 2, 'route_ref': 21, 'name:botanical': 1, 'cons': 1, 'addr:housenumber:en': 2, 'name:zh': 2131, 'tracktype': 64, 'handrail:left': 1, 'protect_class': 79, 'stars': 17, 'sport': 586, 'name:lt': 1, 'construction:leisure': 1, 'layer': 10343, 'plant:output:electricity': 2, 'smoking': 35, 'handrail': 13, 'motorcycle': 66, 'intermittent': 2, 'fuel:diesel': 1, 'voltage:primary': 1, 'transformer': 1, 'taxi': 25, 'building:min_level': 142, 'buses': 1, 'destination:ref': 13, 'theatre': 3, 'toilets:disposal': 11, 'architect': 6, 'building:type': 6, 'telecom': 1, 'contact:website': 1, 'toilets': 1, 'name:zh_pinyin': 1706, 'content': 1, 'opening_hours': 30, 'embankment': 21, 'name:zh-hant-CN': 9, 'name:pl': 1, 'hgv': 36, 'wall': 2, 'iata': 1, 'trees': 2, 'aerodrome': 1, 'hoops': 10, 'max': 1, 'contact:phone': 1, 'water': 269, 'destination:street:to': 40, 'tourism': 225, 'name:zh-CN': 9, 'rail': 1, 'destination:backward': 1, 'internet_access:fee': 7, 'note': 560, 'handrail:right': 1, 'shop': 172, 'fixme': 60, 'mtb:scale:uphill': 6, 'other_name': 2, 'disused': 1, 'name:ru': 10, 'office': 141, 'shelter': 95, 'communication:radio': 1, 'tactile_paving': 4, 'smoothness': 11, 'usage': 563, 'roof:height': 345, 'addr:postcode': 342, 'service_times': 1, 'phone': 64, 'crossing': 202, 'operator': 85, 'addr:country': 10, 'name:ko': 9, 'area': 700, 'name:fr': 96, 'addr:street:en': 30, 'horse': 272, 'sport_1': 1, 'leisure': 2066, 'seamark:type': 2, 'building:height': 16, 'busway:left': 92, 'private': 3, 'highspeed': 94, 'oneway:bicycle': 143, 'footway': 977, 'lanes:backward': 33, 'name:zh_py': 3, 'turn:lanes:forward': 16, 'maxweight': 38, 'alt_name:zh-hant': 7, 'lanes:psv:conditional': 1, 'name:en': 23860, 'colour': 62, 'old_ref': 2050, 'old_name:en': 518, 'leaf_cycle': 65, 'delivery': 2, 'address': 2, 'alt_name': 77, 'shop_2': 2, 'name:da': 1, 'tower:type': 2, 'seats': 1, 'escalator': 8, 'name_1': 1, 'wikipedia': 59, 'place': 22, 'zhb_code': 12, 'contact:facebook': 3, 'source': 8191, 'location': 16, 'source_ref': 2, 'vehicle': 1, 'runway': 8, 'network': 98, 'ramp': 12, 'psv': 57, 'building:part': 3013, 'building:material': 409, 'bicycle:right': 11, 'mofa': 29, 'car': 2, 'heritage:ref': 391, 'religion': 55, 'goods': 40, 'roof:direction': 132, 'roo

```
f:shape': 2381, 'built': 4, 'addr:interpolation': 13, 'passing_places': 1, 'tourist_bus': 2, 'takeaway': 3, 'building:levels:u
nderground': 11, 'nsdi_code': 15, 'government': 2, 'alt_name:zh': 8, 'shelter_type': 1, 'dock': 4, 'designation': 14, 'lanes':
8167, 'communication:television': 1, 'track': 1, 'source:bicycle': 2, 'devices': 1, 'sidewalk': 47, 'sac_scale': 6, 'landuse_
1': 1, 'voltage': 1377, 'disused:aeroway': 1, 'oneway:bus': 1, 'levels': 2, 'proposed': 10, 'indoor': 1, 'toll': 1, 'moped': 4
4, 'dispensing': 1, 'opening_date': 2, 'social_facility:for': 7, 'building:use': 13, 'turn:lanes:backward': 8, 'outdoor_seatin
g': 2, 'cables': 45, 'segregated': 236, 'train': 2, 'motor_vehicle:conditional': 1, 'min_height': 1003, 'operator:en': 4, 'typ
e': 10, 'lanes:psv': 4, 'public_transport': 369, 'trolley_wire': 69, 'is_in': 1, 'name:ca': 1, 'female': 1, 'icao': 1, 'buildi
ng': 28275, 'bicycle': 1333, 'plant': 3, 'short_name': 5, 'motorroad': 12, 'craft': 1, 'official_name:en': 5, 'service': 3221,
'swimming_pool': 1, 'start_date': 368, 'maxheight': 73, 'addr:full': 2, 'whitewater:section_grade': 1, 'waterway': 3952, 'pie
r:material': 6, 'alt_name:en': 44, 'wikipedia:en': 4, 'cutting': 19, 'addr:city:en': 1, 'name:ar': 1, 'roof:levels': 89, 'weat
her_protection': 2, 'ramp:bicycle': 4, 'golf': 68, 'old_name': 828, 'substation': 23, 'name:eu': 1, 'wheelchair': 40} {'spor
t': 11, 'is_in:country_code': 2, 'highway': 18, 'to:en': 31, 'layer': 4, 'alt_name_1': 1, 'wikipedia': 56, 'heritage': 4, 'pla
ce': 2, 'source': 36, 'ele': 2, 'official_name': 1, 'entrance': 1, 'railway': 4, 'restriction:motorcycle': 3, 'height': 4, 'ne
twork': 58, 'building:part': 3, 'roof:colour': 3, 'surface': 10, 'building:min_level': 4, 'amenity': 39, 'access': 4, 'heritag
e:ref': 3, 'military': 1, 'religion': 1, 'roof:shape': 6, 'ref': 133, 'website': 3, 'name:zh_pinyin': 28, 'name:zh-traditiona
l': 1, 'addr:housenumber': 116, 'opening_hours': 1, 'to': 42, 'name:pl': 1, 'old_name:fr': 1, 'name:zh-classical': 1, 'wikidat
a': 62, 'building:levels': 24, 'addr:place': 79, 'name:ia': 1, 'contact:phone': 1, 'name:ja': 6, 'designation': 2, 'water': 2
5, 'tourism': 4, 'from:en': 31, 'addr:street': 43, 'except': 3, 'foot': 2, 'note': 2, 'name:zh-hant': 4, 'level': 2, 'name:p
t': 2, 'shop': 5, 'fixme': 1, 'boundary': 42, 'name:ru': 12, 'name:cs': 1, 'office': 1, 'name:eo': 2, 'addr:postcode': 5, 'sou
rce_1': 1, 'site': 6, 'phone': 1, 'operator:en': 3, 'operator': 79, 'type': 1139, 'route_master': 27, 'name:nl': 1, 'public_tr
ansport': 172, 'name:ko': 1, 'is_in': 1, 'area': 14, 'name:fr': 4, 'restriction': 184, 'addr:street:en': 1, 'name:ca': 1, 'lan
duse': 74, 'building': 342, 'bicycle': 3, 'short_name': 1, 'ISO3166-2': 2, 'route': 129, 'leisure': 48, 'heritage:operator':
3, 'name:de': 9, 'natural': 60, 'man_made': 3, 'addr:city': 11, 'code': 1, 'admin_level': 42, 'motor_vehicle': 2, 'start_dat
e': 3, 'emergency': 1, 'addr:housename': 1, 'addr:district': 2, 'building:colour': 1, 'source:name': 1, 'alt_name_2': 1, 'li
t': 2, 'waterway': 29, 'name:vi': 8, 'frequency': 3, 'roof:levels': 1, 'colour': 50, 'name:en': 317, 'name:zh-simplified': 1,
'int_name': 8, 'old_ref': 6, 'name': 437, 'name:hu': 1, 'old_name:en': 5, 'short_name:en': 1, 'from': 43, 'alt_name': 31, 'nam
e:kn': 1, 'old_name': 5, 'timezone': 2, 'name:io': 1, 'name:zh': 166, 'ref:en': 1, 'public_transport:version': 60, 'restrictio
n:bicycle': 3}
```

检查TAG中K值是否是其他K值的前缀

In [70]:
```
def check_one_prefix_of_set(k, src_set, result_list):
    for k_name in src_set:
        regex = re.compile(r'^' + k + ':.*$')
        m = re.search(regex, k_name)
        if m:
            result_list.append(k_name)

def check_prefix(k_set):
    result_counter = {k_name:[] for k_name in k_set}

    for (k_name, k_list) in result_counter.items():
        check_one_prefix_of_set(k_name, k_set, k_list)

    keys = list(result_counter.keys())
    keys.sort()
    return [(k, result_counter[k]) for k in keys]
```

In [71]:
```
#Reletion_tag_k的前缀情况
for (k, k_list) in check_prefix(REL_TAG_K):
    if len(k_list)>=1:
        print(k+"==>" + str(k_list))
```

```
addr:street==>['addr:street:en']
building==>['building:part', 'building:min_level', 'building:levels', 'building:colour']
from==>['from:en']
heritage==>['heritage:ref', 'heritage:operator']
is_in==>['is_in:country_code']
name==>['name:zh_pinyin', 'name:zh-traditional', 'name:pl', 'name:zh-classical', 'name:ia', 'name:ja', 'name:zh-hant', 'name:p
t', 'name:ru', 'name:cs', 'name:eo', 'name:nl', 'name:ko', 'name:fr', 'name:ca', 'name:de', 'name:vi', 'name:en', 'name:zh-sim
plified', 'name:hu', 'name:kn', 'name:io', 'name:zh']
old_name==>['old_name:fr', 'old_name:en']
operator==>['operator:en']
public_transport==>['public_transport:version']
ref==>['ref:en']
restriction==>['restriction:motorcycle', 'restriction:bicycle']
short_name==>['short_name:en']
source==>['source:name']
to==>['to:en']
```

In [72]:
```
#Node_tag_k的前缀情况
for (k, k_list) in check_prefix(NODE_TAG_K):
    if len(k_list)>=1:
        print(k+"==>" + str(k_list))
```

```
addr:street==>['addr:street:alley', 'addr:street:en']
alt_name==>['alt_name:en', 'alt_name:vi']
artist_name==>['artist_name:en', 'artist_name:zh_pinyin']
building==>['building:levels']
entrance==>['entrance:main']
exit_to==>['exit_to:en']
healthcare==>['healthcare:speciality']
heritage==>['heritage:ref', 'heritage:operator']
internet_access==>['internet_access:ssid', 'internet_access:fee']
memorial==>['memorial:type']
motorcycle==>['motorcycle:type']
name==>['name:lt', 'name:fa', 'name:hr', 'name:he', 'name:jbo', 'name:zh_pinyin', 'name:sk', 'name:zh-hant-CN', 'name:pl', 'na
me:ro', 'name:tr', 'name:ia', 'name:ja', 'name:el', 'name:zh-CN', 'name:af', 'name:ku', 'name:sv', 'name:zh-hant', 'name:pt',
'name:ru', 'name:cs', 'name:th', 'name:eo', 'name:ko', 'name:fr', 'name:ca', 'name:cn', 'name:de', 'name:it', 'name:sr', 'nam
e:es', 'name:vi', 'name:ar', 'name:uk', 'name:en', 'name:hu', 'name:kn', 'name:da', 'name:zh', 'name:bg', 'name:io']
note==>['note:plack', 'note:de', 'note:en']
operator==>['operator:en']
population==>['population:date']
ref==>['ref:vi', 'ref:cn', 'ref:en']
service==>['service:bicycle:repair', 'service:bicycle:chain_tool', 'service:bicycle:retail', 'service:bicycle:pump']
source==>['source:ref', 'source:name', 'source:population']
surveillance==>['surveillance:zone', 'surveillance:type']
traffic_signals==>['traffic_signals:sound', 'traffic_signals:vibration']
waterway==>['waterway:sign']
website==>['website:en']
wikipedia==>['wikipedia:zh', 'wikipedia:en']
```

In [73]:
```
#Way_tag_k的前缀情况
for (k, k_list) in check_prefix(WAY_TAG_K):
    if len(k_list)>=1:
        print(k+"==>" + str(k_list))
```

```
addr:city==>['addr:city:en']
addr:housenumber==>['addr:housenumber:en']
addr:street==>['addr:street:en']
aerodrome==>['aerodrome:type']
alt_name==>['alt_name:zh_pinyin', 'alt_name:zh-hant', 'alt_name:zh', 'alt_name:en']
bicycle==>['bicycle:right']
bridge==>['bridge:structure']
building==>['building:level', 'building:levels', 'building:colour', 'building:min_level', 'building:type', 'building:height',
'building:part', 'building:material', 'building:levels:underground', 'building:use']
building:levels==>['building:levels:underground']
communication==>['communication:radio', 'communication:television']
construction==>['construction:leisure']
cycleway==>['cycleway:left', 'cycleway:right']
destination==>['destination:ref:to', 'destination:street', 'destination:lanes', 'destination:to', 'destination:ref', 'destinat
ion:street:to', 'destination:backward']
destination:ref==>['destination:ref:to']
destination:street==>['destination:street:to']
disused==>['disused:aeroway']
handrail==>['handrail:left', 'handrail:right']
healthcare==>['healthcare:speciality']
heritage==>['heritage:operator', 'heritage:ref']
internet_access==>['internet_access:fee']
is_in==>['is_in:country']
lanes==>['lanes:forward', 'lanes:backward', 'lanes:psv:conditional', 'lanes:psv']
lanes:psv==>['lanes:psv:conditional']
motor_vehicle==>['motor_vehicle:conditional']
mtb:scale==>['mtb:scale:uphill']
name==>['name:pt', 'name:cn', 'name:es', 'name:uk', 'name:no', 'name:he', 'name:ja', 'name:ang', 'name:zh-hant', 'name:de', 'n
ame:botanical', 'name:zh', 'name:lt', 'name:zh_pinyin', 'name:zh-hant-CN', 'name:pl', 'name:zh-CN', 'name:ru', 'name:ko', 'nam
e:fr', 'name:zh_py', 'name:en', 'name:da', 'name:ca', 'name:ar', 'name:eu']
note==>['note:height']
official_name==>['official_name:en']
old_name==>['old_name:fr', 'old_name:en']
oneway==>['oneway:electric_scooter', 'oneway:bicycle', 'oneway:bus']
operator==>['operator:en']
parking==>['parking:condition:right:time_interval']
plant==>['plant:output:electricity']
ramp==>['ramp:wheelchair', 'ramp:bicycle']
ref==>['ref:cn']
short_name==>['short_name:en']
social_facility==>['social_facility:for']
source==>['source:name', 'source:bicycle']
toilets==>['toilets:wheelchair', 'toilets:disposal']
tower==>['tower:type']
turn:lanes==>['turn:lanes:forward', 'turn:lanes:backward']
voltage==>['voltage:primary']
website==>['website:en', 'website:zh']
wikipedia==>['wikipedia:zh', 'wikipedia:en']
```

2.审计元素与字符

3.探索用户

In [13]:
```
BASIC_NODES = ["relation", "node", "way"]

def get_user(element):
    return element.get("uid"), element.get("user")

def get_user_of_map(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if element.tag in BASIC_NODES:
            user_id, user_name = get_user(element)
            users.add(user_id + ':' + user_name)
    return users
```

In [14]:
```
users_set = get_user_of_map(OSM_FILE_NAME)
```

In [17]:
```
print(len(users_set))
print(users_set)
```

1538
{'5226660:Hamed Azimi', '4038854:Chip packet', '2719768:j_c_schwartz', '831660:JJS123', '5555475:Jonathan Mercier', '92274:adjuva', '3872017:Duc D', '4651232:Stevenhuxin', '3649021:Branos', '2433219:Julien Nguyen', '193291:MapMakinMeyers', '6765615:大乐0', '4190070:Fringson', '4140016:Richy_B', '1803172:zenorm', '2348833:lexrkt', '580542:miklas', '5825765:gsw343', '67507:Wolle98', '84069:AfricaTwinTreiber', '4040769:ZFWZFW', '94660:m_angyal', '345866:hob', '5798659:Tommy0933', '364:Edward', '26726:lonvia', '1976819:forsaken628', '624466:panhoong', '2427054:saic', '1189602:Jack007', '55381:spi010', '1219875:Theodin', '1829683:Luis36995', '7406:jynus', '4902267:Will Rynearson', '1877821:greendyj', '3805176:Gareth Nicholson', '17497:katpatuka', '934310:Deruid', '4590461:姚依辰', '6624298:家在花桥', '852124:CyrilTLS', '6447498:CENTRALHUB', '1967105:louisliang', '4645989:lightxu', '4702432:binjian', '2314966:winsky', '632378:3yoda', '1167947:Beinuo', '6603006:家涞坊酒店公寓', '6853928:Swordie', '152289:cgu66', '5269935:大华哥', '504900:alangis', '1897003:joyce1981', '1638106:cnapf', '145231:woodpeck_repair', '4111680:MissiChen', '6608834:張正忠', '7054122:Kaiyang Chen', '2036923:AddisWang', "4825785:Ben O'Reilly", '2284689:morloy', '5100441:williamjoy', '34385:Soub', '4379011:巴山夜雨', '336460:robgeb', '6249321:融融 young', '4091244:Sheikh Salman Ahmed', '69918:0815andi', '207223:kenji', '2329388:canney', '2239961:DerLakaiMS', '4975947:xuhaogang', '3953891:Simon Dumpling', '6586812:wangappu', '154757:Tr4sK', '5350151:dadli', '3722704:asei', '3191854:shaoshanglai', '2504230:Sthape', '1855257:Huang AMing', '1731253:keepright! ler', '207259:endo', '4637640:didida', '236361:NoelB', '3786101:mescolar', '4236594:沉默的上帝', '4227975:Eric DeWitt', '1813711:mr_mizu', '1928265:T4K30', '1778953:mattchn', '374193:bal_agates', '910750:code9527', '4584248:potato1998', '3378150:ChenFF', '173623:JDub', '1897296:loghbb', '527986:zzcolin', '5767812:krystal1900', '599214:East Pano', '40481:Jeremy Wickersheimer', '2135641:matp93', '3691120:samsung galaxy s6', '950182:RitterR', '3365178:Can Wang', '1984668:CaryYang', '13203:bahnpirat', '5167698:Evan Simpson', '2511930:joseeulogio', '4444816:Brucemybrother', '5015997:michaelmdf', '1893894:DanieleP', '2215592:xiaotu', '4071949:Eanass A', '2845965:ZHQ2015', '804208:Dries Bos', '5420221:Buckreet', '4572124:Gilnei', '3892497:ctrains', '1966716:vincentsnow', '5244625:disantos', '173208:moto0815', '1775293:猪刚烈', '646083:freak_antoni', '398086:lesko987', '2584328:SevLu', '2387454:jovian', '422716:Liu`e', '78656:Walter Schlögl', '131670:Ana Luisa', '489096:P Jurgens', '3786977:张文睿', '2120244:嚒lovemj', '6699459:wenghousing', '311833:FabiB', '755587:Schneehase', '7162184:chen_star', '409326:MErh', '4308380:EdwardEngland', "4257325:Marina Carneiro Sant'Anna", '1525242:Richard Chen', '165869:chdr', '2077915:fh265', '5027288:becaycai', '1880253:仰钧天', '1195627:gforman44', '1873388:boldchan', '2084207:Cinderia', '3814947:Adrian Berner', '152074:beweta', '66160:Xylem', '4910177:map-vl-ru', '303079:Sunng', '43807:digdigger', '4490994:Barbara_Dunst', '7156103:Geekolas', '6062216:yannikshin', '529821:fennivel', '4467162:FGAlbero', '3508096:Oz_Phoenix', '3448436:shadowcz', '511967:Besson', '3790564:Juan A', '6602894:simhan', '128186:malcolmh', '2121658:阿泰泰', '6269119:Percy Hong', '584529:autoasm', '458172:Shem', '588693:mayulu', '1824494:EdSS', '3137628:Jingspace', '338611:marek kleciak', '472653:Pavel47', '2401380:amhodgez', '4893896:7thgrade', '2610016:mimiyukii', '3963214:LibbyBell', '3497647:Han Long', '3942472:Julian Eberle', '2445888:net I net', '2163975:hqg01', '2835928:nikhilprabhakar', '117119:se_osm', '4860542:JiaoGisEcnu', '1597155:poornibadrinath', '392827:pnorman_mechanical', '5731031:Joelee71', '333196:Robbin Wang', '2704928:Bareman', '582611:trackbytrack', '2142148:yangqinzhi_cn', '4973921:michaelinshanghai', '320455:kgbkgbkgb', '91490:Heinz_V', '5316306:林加志', '6044384:DenisYu', '384084:Kostik', '300544:japuifor', '6600402:Li33a', '6149343:Denis Ding', '6095480:Tony19708', '4004818:liujy1005', '4325719:Metro Juze', '5358888:Dawei007', '11463:plikma', '2361109:wzlh007', '4949583:智向阳', '1873836:Erasmus_Huang', '5248074:wanou', '3859534:csongl', '647416:量子妖', '4467965:Peng Seng Ang', '5230141:Nehaj', '17490:Adam Geitgey', '51722:Chris Parker', '2186723:AlexWang0315', '6124081:Kristina Khairova', '7028067:kkGEN', '7119508:KingCapri', '208047:Basti der Entdecker', '2834413:Alex114', '1836471:Blobo123', '3163758:jiulongjun', '94578:andygol', '3642735:Nodes&Roads', '1030836:Artheru!', '4628193:gri-is-real', '31231:Andre68', '1016648:zhangxinjin', '2507347:中鲁潍昌', '2085910:love兔兔喵', '4864276:_eder_', '3007309:thorbenj', '5432507:Knockerclot0715', '510836:Rub21', '3019749:Ping Chen', '1929621:ttcc', '13257:rolandm', '7139407:Dave Marco', '2128793:LinuxNerd', '4892790:duknic', '3304372:caogaoya', '671467:amour', '3320809:nanosun', '162827:Hb-', '4501118:Parabonaut', '1936164:fantasy913', '38553:egon', '68661:PeetTheEngineer', '2086028:yatee', '3739650:Dubhe', '4102363:Jole22', '4316231:Suky Ting', '4505853:Josetheexplorer', '61625:ghh', '3325068:reichsmark', '83557:Esperanza36', '210430:Barnabe', '4227826:Jorsh Gonzalez', '7124352:Silvampire', '5502068:SinggingCalf', '28559:Stemby', '188811:fall', '3618297:scac2015', '5547005:Shen Soon Chin', '5810704:Rachel McCandless', '6116235:Frankenfelix', '2100300:shulinpeng', '1911765:lksl', '6272015:ayitaiyi', '4455892:小头爸爸', '2390101:HYLuuun', '2385757:gekapes', '129772:smchadwell', '647232:tedu', '2704394:JojoMonster', '4905628:Santiago Trujillo Pereira', '4588941:Jerin Mathew', '6410877:zhuchao110', '6664188:Jonhen Liau', '4820245:Victoria Romanyuk', '5506718:钩钩小指头', '2947535:史小姐', '5499990:adrain728', '2952446:Virgil Guo', '6508127:lonaliisa', '609950:rita xie', '499500:hanchao', '1082589:gide87', '2376:Bman', '2275300:hhhmmm', '421504:u_kubota', '596114:z_i_g_o', '6739985:Julieve', '4899900:Alice Ruiz', '5292771:沈轻舟', '1780757:XU Zhiping', '5589870:tymwqh', '5597502:Дмитрий 61', '97488:Alex Prince', '1966673:lamb_da', '3974890:Frerix', '64922:ulrichsommer', '110639:aighes', '3250961:lrnzmnnl', '4368807:可可西里', '2891170:alberth2', '4877996:zzslamzz', '4206342:Pierrefleg', '5142157:minaolenGlu', '2889758:Cccmm002', '4838252:千岛寒流', '23644:Lujason', '24359:user_24359', '1739316:CLuo', '402277:sedas', '448087:schumyHH', '103253:gormur', '2899929:LJun', '39826:trarbach', '375745:kingston51', '5355619:Maren Volden Smehagen', '131968:changchun_1', '366906:M G Jones', '6471337:wwwindow', '170106:sanchi', '4995942:jihye choi', '621514:Shangdaemon', '6246267:robinsparkles', '4731522:KelsiS', '5560958:barefootrider', '6274368:hgellert', '2013615:redbuta', '3292947:vendetta689', '3316592:Arthtoach', '3800648:夏赟珅', '616944:andreas153', '27454:Popolon', '2085913:jinjinlanfeng', '445671:flierfy', '101611:JoergB', '109970:mpaeslack', '3876852:drunkensailor1700', '4339167:OneStopNodeShop', '2440593:annaliu', '612380:k_deboer', '6144754:LeDuys', '3579215:jiehecn', '6215939:瓯南交迷', '1207628:Torobucks', '1893529:manits', '3840641:星空之梦', '70952:ThomasSteininger', '35949:Andy Wilde', '2124807:feb2014', '4601320:HMCho', '2223971:躺着看云彩', '1240849:ediyes', '2381605:AUGUST0414', '3839010:JT2016', '1062577:kennyG', '2188716:strongwillow', '6716597:caixingyu', '611919:jeniffer', '2364671:deeperthanpain', '3721989:TomicWhite', '2258307:como un burro', '2126692:508', '2084110:jpmapr', '4625008:Xiaopa', '279576:Beelsebob', '6865760:Kempinski', '225559:TZorn', '2019423:cutemaps', '4109125:P-Hat', '16619:Thomas1972', '4511896:wei xu', '3187712:StellarStar', '6394980:personli', '1899322:SH小头爸爸', '1907534:dermirt', '2209900:jijhy', '5028199:one35lab', '2551002:逸竹山人', '4635042:Onthewaycn', '2083375:vesor', '406626:王大可', '3784677:地球小兵', '15740:Piet Stevens', '1413017:Kashif Alvi', '4857423:wreger', '3316212:alex10a', '2222654:pyenney', '2717795:leezhihong', '640614:dfxbb', '4422697:Geraintosaurus', '7111708:jackforerer', '63107:brogo', '1795336:homer__simpsons', '2109207:ellenlyj', '3935135:Cornelius Neumann', '3839858:sonia_shi', '4236469:mryin828', '651869:Aurimas Fišeras', '4846907:Enrico Odasso', '672536:xmd5a', '445743:ac201', '3339609:Frank Xu', '4697764:judyzlf', '2748195:karitotp', '6240815:howard_zh', '211978:rudgerle2', '3713193:DrWhite', '5380436:Luciente', '1502427:MatteoFrittoli', '3335289:IMDommy', '52797:hofoen', '5595757:十一yi', '274857:Supaplex', '718923:Yourez', '3522215:Xiachu Ma', '618789:Leouu', '4190383:Phoebos', '3115220:RemisC', '5875751:lvjun', '1919824:none name', '6116532:max328', '3323283:sjdd0001', '4058516:Cash421', '1509982:nycrare', '1617449:aceman444', '4561463:harulf', '3528520:birella', '3851347:Martina W', '460650:tommy_xuyijun', '3380938:Mickey Sun', '1951111:FrankGu', '5553:dkt', '1887977:revent', '628203:Sah Lee', '2535907:tular', '2618832:刘影与自行车', '2508151:ridixcr', '5368687:yege', '1804331:daxigua', '24942:mannequinZOD', '717089:foxyflash', '42123:Ropino', '298870:wayneu', '3124055:Workoft', '4721605:CZhuo', '7213:alimamo', '6769096:kbitr', '1314388:Leo Wang', '4751085:TrrZ', '665748:sebastic', '2034654:nemo17', '4906777:ssxs', '1073950:William Denniss', '3882811:porwalnikhil', '646907:LaurenceXu', '126036:has4', '4289312:philhan', '697073:gergl', '6911076:Ryan09', '6339017:Krelf', '6069860:Mapperhugh', '4629299:Fumihiro Fuchiwaki', '2708031:geluadi', '3459524:hegxiten', '4497726:Bo22690', '3790307:Iamanenigma', '224918:甯小塞', '3297680:jiaodalpp', '2856209:Daniele Fissa', '5724531:经冀Hostel', '2085058:KKKKyrie', '170289:mAtchpOInt', '4749825:klemtors', '4632761:cuipengxia', '1306:PlaneMad', '5744474:СК СОКОЛ', '42429:42429', '1892020:brgr', '481533:dbaron', '8339:mtmail', '4182624:유미미', '18052:sfrank', '4216484:archiechak', '3963148:Feng Jin', '2594717:VictorXu', '4768459:Global_Player', '6421363:chj2017cn', '200922:hfyu', '2009091:onizuka0440', '635876:henry135', '6182776:Than Soe', '4578254:JiaGIS', '6154898:Marcin020787', '3576192:zhaoquan', '50253:Stefan-China', '5048345:hdm1000', '711265:irasesu', '83411:jduffhues', '1420318:4b696d', '2913572:chilam2012', '1329617:Martano', '3765004:sakura0608', '604999:White_Rabbit', '9176:Maarten Deen', '2929244:penguin_the_dawn', '1735613:Muchan', '3119547:Christophe9791', '181135:Manu1400', '555014:John Shen', '2747931:Hongkers', '2743989:Ted J', '6387519:yoake', '2159247:Phil Qiu', '107257:FrViPofm', '5333160:闻一多', '811230:Andrés31', '1236135:FreedSky', '621082:Xu Chuan', '6893063:fc0923', '5636697:jjmartin', '4225427:noviceGis', '641843:bladouze_osm', '2225546:andywxc', '6609108:mgbondarenko', '6746972:Alix60', '1310930:shirlyxu', '164748:Matthewmayer', '2431429:cybermix', '568217:蚕路科技', '4808589:Gasto

n Bonaudi', '3922813:RSJmap', '3191182:angys', '844850:DerCut', '6127520:dar44', '5211:emvee', '352940:pieleric', '3162230:yaranaika', '6722:hannesj', '5755486:Salim Pokun', '2249643:XiaoLong Shanxi', '1757906:chengyuan', '1056611:openstreetuser', '5635:casio2000', '4142535:Ruslan Vagizov', '4397687:DSB14', '4808486:AnHu', '2068925:jiangjinliang', '504344:Michael Zhang', '4999581:zhuli0905', '1195808:kun3721', '131048:mongokongo', '700748:jaspertchang', '5740906:TS11', '4510926:goodmapper', '75424:nuklearerWintersturm', '93285:skela', '888376:chinhsuen', '2996489:rheins', '3360:RalfZ', '2992037:trecht', '4595019:kongjiandaolun', '3814758:Abd-Samad Habbachi', '698504:patrick73b', '3528573:ulrich_', '4404650:TAUnionEd', '3663507:jiyeon lover', '4178003:Calligramien', '228662:HGSandhagen', '4908104:only4dragon', '5152867:热血BUG男', '1704882:Yao Yuan', '5509234:Ben97', '341454:MA_Mapper', '1798312:hj132865', '3900861:pierre1011', '1238175:LifeEngineering', '4911383:sngy sngy', '4966582:JuicyTruth', '4855870:Garnitur', '5461695:FelixSH', '1487220:yhilan', '7120051:anthropologist', '4078281:SJS-BIKES', '6727898:e982happy', '7749:Matze', '61891:stephankn', '34124:Sunny', '3800658:爱蕾蕾真是太好了', '4816761:Austin Zhu', '6113658:Alice dutronc', '4673612:xawyzen', '3508588:地图兔兔', '2084339:孤独症患者花花', '4861140:Vanish Sergei', '385027:Ori952', '3112503:Johannes Bunte', '6300295:Chatchai Wongkhom', '68982:kisaa', '3835675:yezi', '123440:nodust', '4545397:Lev Nezhdanov', '1826106:dancingbear', '2767104:novate_z', '3782269:Павел Комаров', '4633931:wudaofan001', '1090704:starlightliu', '356014:_sev', '39504:malenki', '5951885:brad8610', '6233814:GinoZia', '576943:Jeremy豬鼠', '1851374:w4890888', '1783785:RObst3r', '701297:erjiang', '34964:latouche', '1915697:Hilton Hotels', '2247082:up34', '1591520:lbsweek', '2530118:四国军棋军长', '2115749:srividya_c', '3756906:seven_emilia', '1293194:nevw', '2257981:moonofsoul', '3040641:JJFad', '5380652:SuperUser1', '609469:kanvii', '5079977:守夜人', '4523400:InitialB', '3965122:Offhero', '3491310:flashyhl', '1841996:SLussier', '55048:Funbo', '3806526:GeoThings', '5247274:Alain_L82', '6426976:Kelphon', '4909356:moein110', '4726227:Martyn106', '322785:BCNorwich', '4215521:吳奕廷', '4600936:huangzi', '15908:cantece', '257555:rene78', '64557:qpeso', '3926024:AC FootCap', '2219985:ethelmermaid', '3868198:cumbop', '1161938:vampirefan', '6586599:Hugo Fanaia De Medeiros Somera', '5461188:Cverbost', '2286161:daiquping', '2955660:4fatal', '1836406:_rockmachine', '3802669:Jose Ramón Rubio Moldenhauer', '2236799:ztplains', '4320259:Ign24680', '3887052:MELJNIK', '3924694:tNickel', '1486603:youjie', '6364911:海上彼得', '829369:Alard', '5147743:Jason Roy', '6688316:Takashi Sugimoto', '360397:DAJIBA', '2015442:georgeshanghaim', '5680771:熊文嘉', '2081928:pingwang0120', '2268476:flywi', '1722904:Alan Trick', '6469244:ChaniLopez', '5430063:goodtree', '2554698:ruthmaben', '6960425:poplfizz', '2973031:edtha', '2251986:Azca_', '2541036:Devinl2', '110263:werner2101', '2255641:ShaunMay', '6489902:春源散人', '233376:Caba1a', '709001:Alex Clazrey', '3836466:Koalberry', '6416288:刘格好', '697239:HWST', '107148:wdmanuel', '767930:ericmetro', '3912463:GABINO VERA', '6237468:KeesdVr', '3796228:Andrew Chow', '4950482:茕茕白兔', '6351705:Dustran', '223584:rlech', '3787764:nk nk', '2414617:LMilard', '93315:fume', '596047:Rps333', '4159619:freddy_fu', '429332:keydream', '218651:crenz', '715800:ppenguin', '3400070:Joanna Wang', '2214753:toyac', '22827:ouleyang', '180541:chinozzy', '6463522:LinE93', '40984:anlayne', '4737026:Alexandr Zapreev', '3198900:charles92', '5057155:Jerry Hu', '121036:TheFish', '4730624:Wouter Serdijn', '3640254:imthefrizzlefry', '1049537:Minerraria', '4743898:tubachris tubachris', '70170:Berobispo', '5317567:James Millar', '4397792:Petit Lu', '1071456:rmentat', '693098:radek-drlicka', '416346:Brian@Brea', '4796082:David Rondón Manrique', '2004309:xiaobao123pppp', '3782098:bear_ukraine', '3775202:Speedy55', '3715087:Herviet', '3886711:ILikeTurtles512', '3475228:pengyi', '2423635:vimtseng', '5716221:eugenm', '24247:Vlad', '1817244:Helmchen42', '5080040:maxagogo', '4084401:Poetini LEHARTEL', '1816511:xfang2', '6218061:dblndr', '339917:Hedaja', '6464157:桃學匠', '1967648:Bsdos', '1003129:Alex_Shan', '1833906:haoyun', '1292441:lynn zhang', '2759372:Joeyao Chou', '5768895:Roland Born', '321042:species', '546735:Kyuu', '1970556:DerekChia', '1416113:Rick Feng', '2232599:Blade Z', '2095783:alec_r', '1247041:Essex_Boy', '4799088:1102112', '4305288:Samuel Ekakurniawan', '1793309:danielkok', '4692843:Pranav Sethia', '160949:eric22', '74063:yongliu', '2916278:leamhsi', '93757:Tao Nan', '6464150:Stella515', '1819191:Fashion-w', '4336011:MalikJacobs', '1747413:waterfronts', '3926950:Vasselin', '3414196:babribeiro', '2788553:wushana539', '8703:rolandg', '624617:cranecn', '5449774:龙年农人', '2925397:WesleySu', '2085906:碎碎念碎碎觉', '342402:FDF', '3234238:4SM', '3392894:melay', '1984381:mars22', '360392:maxerickson', '3477164:九州昆龙', '2356322:Yuchao-TN', '3220827:citykam', '2238844:姑苏小恐龙', '5397234:JoshFrosch', '2286509:daiSG', '2921845:liupangui', '374577:zpeng', '6783113:Xu Jingyi', '1254612:bensun', '45027:PA94', '619614:Conqueror', '1891299:QFA7301', '2329819:kenzla9', '1790827:Ali Yang', '3412904:lvtianzhou1984', '307202:alester', '4201552:a517817051', '3510066:nebendrin', '6860357:cpkxf', '3283745:frqc', '2202560:uug', '6788677:sbmboy', '4892637:백이슬', '183579:simonb', '1855271:Ken Yao', '2100550:逆风飞翔', '1640335:The Bludger', '6141218:Steve Ma', '1737608:ForstEK', '5208433:Hyeokil', '6179684:pitt hu', '64578:Serpens', '104808:Ic3cAk3', '4688437:human890209', '3264783:自由分享', '48369:Scotty-NUE', '5070034:Sighofiris', '5707824:Narfik59', '1297:garaolaza', '640416:talktojosiah', '2404376:uwaterloo', '1414919:Charlie Loyd', '195610:shatin', '586175:lorkhan', '1730070:riczhao', '6486297:Eleanor May Beresford', '3618977:illya2400', '4777087:Bernhard Schoof', '647420:Roy Shen', '4330149:Lcspl', '4320061:mapszzr', '4458273:Andi Kloos', '386131:zhongguo', '1436097:Holywindon', '26599:cdavila', '207745:NE2', '6058083:地图Q', '396600:Tom317', '3883104:AliceHasAGun', '3189456:Arthur2e5', '960774:whison', '720609:Vulcanodong', '2873697:molloyda108', '4955535:Владислав Вартанов', '623011:etolew', '2754552:Dannie Lu', '1825040:schulf', '2120219:rayeaster', '993315:triplemac', '1738216:adoggie', '5170345:Vancouver Pearl', '1164:dmgroom', '13721:wangchun', '185804:mongbei', '722053:jefaure', '1891981:LazyMuzi', '119281:RudiXXL', '430756:u326875', '6760798:Fabio1979x', '4243584:shimooka', '2328491:Alex_on_the_map', '42039:MatzeR', '5501955:赵宇zhaoyu', '2159337:Tomas-chy', '4892725:luotangjiji', '4293701:vip_vip', '647047:jiji9081', '4642482:呀巴巴西', '616774:mueschel', '2085822:111', '2429841:Siddall', '3222751:kgmaps', '4810806:范春波', '4349815:冰糖喵', '4580497:milopascal', '5891:Accurimbono', '333194:happy000', '212575:muaum', '267767:Goderic', '1263561:wanderingchinaman', '5517259:Ceciliatyou', '2238048:ethanchan', '8609:ewedistrict', '6840268:Leif Cheng', '5687012:菜根谭', '3302953:ChiPJ', '456555:panyf', '1708958:Nodes&Roads', '4751744:Peter Neuhaus', '4957750:Xofaway', '4760005:TeaTime13', '3308004:io-node', '907975:HUANG JIN-CHENG', '5548062:彼方桜咲', '2103533:raynhard', '7010:GarryX3D', '1417804:adong33', '989747:kaimai', '298472:nuy', '5316523:Green Liao', '4194946:Gregg Ritchie', '566149:Zolo', '4844206:gtiq2000', '500935:hicom150', '1869781:AndiG88', '5343976:Silke-augsburg', '3284594:immanuelzhu', '123123:azuazu', '2523109:mariannair', '550560:Seandebasti', '4408550:Charlie Khlaklang', '912799:Cong Li', '1714999:Papa Fletch', '14002:Gehrke', '4070201:rogerpms', '503258:xinzi', '534149:lurlrlrl', '5799999:Vincent Couderc', '4158641:Queenie Chiu', '911575:yfyamato', '680736:frankcz', '566296:wang xd', '622435:alan wu', '201918:STVictor', '5671567:chenkun', '512905:jefo', '3341187:Pepilepioux', '1069176:landfahrer', '5234861:Many9', '4576433:wayne7', '361503:uheeschen', '402306:GlieseRay', '6272023:Daniel JRDV', '663804:JiaLU', '3994312:Valenty Sun', '4833691:bujiatian', '683648:Shuhai', '598819:marineyin', '2014242:PRinMass', '65419:Rigless', '4803512:kevin Prince', '227492:foop', '4737250:kingsir', '4728774:Ranjitsingh1', '2357353:sonnar', '4418199:lubancafe', '2214848:fdulezi', '2801734:elightGIS', '606416:anttit', '4158505:StefanKlaahsen', '2171667:alahover', '4969120:Eddie Seung-hyun Cha', '161798:maximeguillaud', '3250154:kyan18', '6454225:RetroHomer', '124617:david_dai', '5581919:iamsarahj', '5684512:Veri Predi', '514683:lukys1', '1982032:等车的晴天', '778395:quenna', '857408:huting', '1607555:wenhao', '5084475:Tom_bot', '2501310:GeoservicesFDFA', '3490410:bluevoice', '6777092:nanhaiyufu', '3571864:Squideroo', '877337:cdfz', '6225072:emmmmmm', '1016099:fever365', '2073608:yingxin935', '177389:Garmin-User', '4717592:nulandjp', '306409:Skiper', '4361066:ntds071gly', '6554882:headton', '4486641:miomoimio', '1834298:scmast', '4180378:Rafael Lara', '204590:Elron', '1091492:lcencetti', '3450290:ff5722', '655849:Tang Zhewen', '4584812:Camillalala', '26299:uboot', '5328554:Liviu Macovei', '2092749:Weber83', '1427884:creasplei', '3999864:Scott Burkhalter', '2226313:JACK728', '4355902:sghiei', '3164012:danidoedel', '210173:osmmaker', '5286834:Ger Alfaro', '6723807:Daniel_Ma', '1380023:largescale', '5271811:Leonidas Stavrakis', '324185:kimh1', '5294536:DHus', '4592:Mirarkitty', '2377377:abel801', '5136178:TimeGIS', '672542:Nearo', '3693002:GeoCrazy', '6673976:Fang Xiao', '3383242:汇图公司', '1874580:pandong', '2151906:fants', '648577:javis', '4687693:邬浩翔', '3588086:DavidGuo', '6950396:Ryan Z', '1810601:shanghaielaine', '7170928:shifei', '101884:Antwelm', '5363463:SHIYA2017', '475943:sir-mapalot', '5471606:7879769', '644218:LUNAaw', '116515:mcdlee', '124788:Maccy', '2249743:Marc le Grand', '243496:lenzai', '2263439:wasm002', '4293887:Betasy Don', '290680:wheelmap_visitor', '3818966:Thaspong Sirichotworagul', '161619:FvGordon', '4933106:CedricZ', '4329555:Nathanhorn', '5352632:RobertaZ', '4456969:lisamausfan', '4617685:Tayma y Churri GMS', '3718368:抽妹儿', '12269:Kopfgeldjaeger', '664653:Jeremy NG', '3915237:Jim Liang', '5715:Hoylen Sue', '1897202:Anny_han', '1018672:FlyingXiang', '2511706:calfarome', '56597:Oberaffe', '88347:malcolmfrost', '4812456:otternonsense', '3433249:Mr WAN', '6161558:carin_l', '4413239:S-hey', '4654673:Fabrice Zibel', '577952:Sean518848', '67862:HolgerJeromin', '383625:cuas_mapper', '476483:lorenzo23622', '142205:skobbler', '4375109:zcs1786', '2263387:arrowrowe', '3057995:oini', '755059:kingye', '4419811:沈彬彬', '6564339:小黄帽', '1469704:DougPeterson', '546832:kitmention', '2238394:qingmeitekuai', '5173844:ndrw6', '1870508:zhouxiaobo', '1585301:yhz1221', '6105827:WooYoun Cho', '1863966:syuganmei', '3779486:ezrick', '3521219:Manu Stefan', '3929046:BlossO

z', '5095743:Domi M', '4098562:hyson710', '6904824:周海龙', '436419:wvdp', '5030592:ethylisocyanat', '6363348:Didier Varón', '5828909:Calpurnius', '5357015:Conlon', '694419:guoliulong', '535134:Danrvt', '1227933:夯船长', '559441:whx', '1817984:941', '2108065:小小鸟', '6582581:shjanken', '224440:Syl', '722137:OSMF Redaction Account', '3333854:sws12', '4754000:Игорь Наливайченко', '4803527:Kate Diaz', '1795775:christyg', '1051550:shravan91', '41458:bgrl21', '5763246:Dani Berti', '3437781:ti-lo', '3298268:novanova', '597625:sjhbcc', '355855:Yide Tu', '2106440:subtle-fox', '4440132:quax76', '2109169:isaactl', '3356334:陈子峰', '304705:yangfl', '632124:duxxa', '1704117:cyiping', '4514278:Bandits', '579421:EnnoAnwr', '190389:HHCacher', '365951:kuass', '1679:andrewpmk', '560814:afetser', '2210839:Cauchy_Wu', '4206228:Alanatee', '5577080:宫剑辉', '5459943:Kenny Wang', '6413186:MandyMoslow', '2087258:英子了', '580978:flintthuang', '1946823:LinkerLin', '4707528:Isao Yusaku', '4106070:Cladors', '3618976:ProjectThis', '222391:gummbahla', '5432323:Daniel Rutberg', '347944:matthewsheffield', '2545951:highflyer74', '2113287:xushize', '2175648:Emmece', '1503302:Michael Lin', '2125878:MC1200', '2249499:故事宝藏', '1735913:sinnlos', '533465:GerdP', '2004100:whatcrazy', '3218579:bigkopy', '1022650:Jay Sun', '6085327:Tobi Löffler', '497881:Wim van Dorst', '507276:jkjkjk', '1671116:rabbyzero', '23785:greecemapper', '3725439:englishpatient', '5094762:LeadStarian', '1791215:lphu00', '605538:wkh0000', '4912648:BrentHB', '4366968:Max Iubens', '4128942:stepanzel', '5260094:JimmyPL', '5449391:Patrick Hoareau', '1951271:Qiboua', '644783:inlooke', '5369465:Joy的池塘', '2326558:灵犀指', '2219338:RichRico', '847434:vivianjia880127', '2124439:Man77', '1693568:RamboW', '4925517:甜菜123', '251543:ChrissW-Rl', '4609991:shikiiiiii', '908746:dulix9', '1549957:NicoKr', '102398:Sébastien Ferry', '6722488:Rudi Fidel', '181561:mash00', '447933:fetzig', '5672698:Kris Kros', '3339203:Kathleen753', '3327036:wanderlustig', '145520:km_01', '3219846:SunDaDa', '2083307:fatman13', '2819407:Micheldlg', '6362406:alex941338', '1539353:keelunt', '481116:bigalxyz123', '2416145:trumanus', '3284852:mouhao', '4329713:bonylu', '545191:miephos', '7053741:Windst', '4891216:Les Pétards', '4797065:shrimpfield', '4283730:HenSolo', '66811:tilsch', '4260390:IbnTěšfīn', '6289433:mindyalways', '329250:grelin', '6275750:Bingbin JRDV', '51045:Geogast', '1198074:DoubleA', '6789562:AlonsoCF', '4967115:fullike', '1915680:北巷猫眠', '4963152:Trung Ngo Van', '219348:strixaluco', '60345:Mirko Küster', '2226712:dannykath', '1857400:qiangbing', '282922:surveyor54', '3394707:MapperTian', '207581:Hjart', '4716432:Ana_Berlin', '1418137:Test360', '6933701:Wang Ray', '2061320:小珠珠', '666807:jayasiwnd', '454589:sabas88', '3424135:SaraW', '36821:Frankenwaldläufer', '6882820:Davlyat Valiev', '1871670:Eric Abrahamsen', '2937899:Terry Zheng', '624068:koki shibuya', '1328943:Qian Yan', '4846770:イケワキタクミ', '3923956:Guardian Celte', '6713954:施芳芳', '931640:xxcool', '1073593:xuhua', '5467201:Archer444', '1227959:lmayor', '2006304:tony124105', '4363555:Pugsbrew', '24440:jaakkoh', '5451060:muyucel3', '491132:yongxinwork', '2777054:city8', '1185631:Dmitry2013', '2341226:Agora Space', '1889317:nicolajin', '6375918:Columbi4', '6071292:竹叶青上海', '6304168:benechen69', '1294163:affsarauf', '4035796:zhaokd', '4876175:XInLi', '4869184:laigang', '505433:homerlu', '6053109:pingdan', '4811931:rajshrestha', '4172387:tinybox', '1828230:Rain_2011', '3581489:yewen', '3892981:SARGve', '1879371:johnleehook', '520239:Gutsycat', '3489345:xgzpan', '6069836:SBry', '172061:CloCkWeRX', '602634:matx17', '2114252:EricDiao', '251848:Flood', '2880181:Betzilla', '6546783:Future R', '253683:sinopitt', '4344145:Chizuru Kobayashi', '5465124:Vanessa H-b', '166129:BugBuster', '2963470:SSYoung', '5441950:KK_VMP', '1929581:thinkingtime', '6087605:Charles_der_Geographieschüler', '977652:darkofday', '4252168:Sungwen Yen', '2074433:siumo', '3404001:TOOOMATO', '1777257:landygu', '233759:TOBi', '5728414:Tatsuaki', '21118:miurahr', '6464312:人熊猪', '69102:Mathbau', '559500:johanemilsson', '4041099:Cale_xox', '775666:gcjunge', '131474:MiBu', '5849112:Sharon Avisar', '2009829:shiwei', '2500:jamesks', '6852933:DimLobax', '5315592:暗恋宝贝', '5686160:전수원', '3362769:SalsaConsul', '1128933:frankde', '4842543:Z_celine', '2387007:cftm82', '5359:user_5359', '2644101:Chetan_Gowda', '2391970:sical', '106914:Hobgoblin', '1670224:yifan918', '179581:Geofreundl', '2232620:Ethan_lay', '574907:张校玮', '4227941:chirosempai', '6180468:Alex Farthing', '2006256:johnnygu', '112189:cucu', '145401:jnstahl', '4851880:Isaac Aloysius', '3232058:destec', '4497612:tluyben', '6085223:noelschwarz0906', '5230895:a_zh', '603325:geofoxx', '905716:Guylamar2006', '4023968:DagM', '3497015:AntiEntropy', '6280472:la vache', '618287:georgesandlily', '975150:Pau69', '2512300:samely', '3813456:Марина Марина', '540071:bakasana', '4996218:서창성', '1873071:kiolou', '4835302:Leo88', '6114137:Mauro Mundo a Fora', '4363:Pmz', '6087:Geonick', '2184890:edwinlee', '5125558:Rodrigo Sepulveda', '4117507:NicoTho', '4595947:ilovesxl', '907874:tianma', '1093786:leo xie', '4206924:Stallionaire', '4077903:NStyle', '910740:崔 成浩', '6750028:Tonnyvdmost', '313327:Claude3386', '3076241:CrysKrow', '5221710:Stone Cutter', '4618979:Robin Zhou Rui', '80412:dasy', '1837130:arnat988', '4450909:raulbeni', '5310225:polar-bear', '4867371:Toni Monner', '3241388:gilbertz', '3775988:Peter Feng', '42546:Reinhard75', '49873:TooM', '1917502:zhifeng', '4803528:Eva Blue', '3579630:郭唐磊', '4643348:kydcao', '2125554:chong_o_0', '3743231:Luluscape', '2649006:Mappino', '3998564:Андрей Маслов', '2279682:Tomatohed', '5594803:meng8131', '715998:xgongx', '1862296:00PAM', '6480964:edwuwin', '7120722:kanaxi', '554973:dgitto', '1962477:jc67', '4156387:fhiutc', '2263256:自己的地图', '2195990:torkx', '756:Mirar', '3796865:Leo Sun', '5674340:sean cai', '142262:Gerard_De_Kremer', '6575871:孙汇洲', '339581:nyuriks', '6824487:hatelove85911', '18069:Claudius Henrichs', '395756:FrankVD', '166235:GoranC', '577063:QBox', '6326:mike_hd', '1807278:gameinsky', '717081:Frank Zhao', '6640978:Jose-san', '3691003:Garfield99', '365274:thbz', '4428871:Thanos Tsitsiflis', '5828317:AlexFree', '2757603:Free Mapper', '2944142:hpsun23', '1891561:jiangminyan', '388809:市川ゆり子', '5047479:Enzo Lin', '213293:djsimon', '813560:guddi2008', '6804112:Patricio Vidal', '204045:Toliman', '3521744:starsy', '3921465:yearsye', '6415656:AndyHuang_CN', '327066:kekulik', '1960708:hnngm', '2547999:我不吃大肥肉', '3386984:chinaoaxaquena', '4087230:Рустем Зарипов', '228596:panoramedia', '4965125:Eric 888', '4070521:Shaurya Jha', '577816:martin0203', '4497723:ahab003', '1420035:zyrope', '703046:liteng', '5238319:StanMaps', '220682:minourii', '5231023:5714206', 'כרובי:Felipe Segatto', '4368579:Shangkuan Lin', '654517:hcl67', '4253172:Franky_78', '3995433:Dirk Tiedtke', '5496247:JustynaPatrick', '4982682:xbd1519722757', '1947333:liushijiemail', '6656931:she xiao', '2620510:tommydragon', '27577:Depsy', "5782725:R'Kawai", '1992246:bobosui', '603131:superx7', '648195:Jean-Pierre DAVID', '6486491:杜卜荷', '4803525:Shaun Austin', '1846027:trickysky', '3306:Russ', '275821:Ramzes', '214969:AndrewBuck', '694726:aseerel4c26', '85218:Aleks-Berlin', '4693585:mAcKeka', '222066:fxx2000', '201024:WMap', '1190212:Moovit Team', '1967021:ogosoh', '505667:Bullroarer', '193401:photoluc', '624020:bigtigerlu', '5719572:mushugan', '4396547:HebeineseDoItBetter', '4846445:Václav Fanta', '33757:Minh Nguyen', '18106:geltmar', '2161982:chenyongbin', '2318:Latze', '3391896:hpjsh', '5067406:edyu', '70696:xybot', '1978817:neusung', '382514:zone', '87483:Feilipu', '5519414:asien', '2492274:hardcorer', '1122708:zaizone', '443130:Alan Bragg', '6078275:Raúl Martín Félez', '3920007:pjolicoe', '413914:KartaBY', '714980:Nicholas Fong', '4515447:Sidonie Xie', '4835500:SJHXJTU', '2148110:牛嘎悦', '5381276:Muriel Janke', '6464148:梨子yyy', '4455219:胡英明', '3374518:wmgleckner', '672155:Mangolime', '485866:foerotpz', '5769897:juejove', '276615:krafttom', '666915:wilson5429', '351532:tothod', '4916390:Andrea Tosques', '2108064:陈小鱼', '378737:Scrup', '4589542:你妈说', '150368:Cicerone', '6512748:MWDK', '399780:Greco Shi', '4859357:Troy Tang', '5019704:Siwen', '4757277:taketaketake', '967832:Kurly', '2057798:Sarvihepo', '2090620:zjy7770', '1443767:Olyon', '1133244:Min Zhang', '4448511:jianglisheng', '872239:Tcharvasky', '6995990:Nakakoue', '2050751:tedwado', '118021:maggot27', '2269211:chelouche', '652570:meiti', '527721:ninjamask', '4601309:阿飞积极', '4850471:LeslieWong', '5160273:Lit90', '2795189:DavidPoppell', '2088610:xsy', '3282625:Atanvarno', '285891:Jango', '1778799:SomeoneElse_Revert', '5017860:树精灵', '3820373:Too Hot To Handle', '1483661:杨志宏', '3392164:才不告訴你擂', '5380636:Im Watching You', '565800:Ruiqi Wang', '3019495:Kim Shanghai', '6835221:thumbo', '4002381:Nicole94', '1234254:xdpitest', '2445224:rowers2', '652913:Haaninjo', '4858636:hmyjbd123@yahoo', '2217888:曹启承', '1872917:thisiswiki', '5672132:nahuele', '4784221:Kolinko', '4354162:nataliko', '1713646:张正华', '4051669:loconvey', '4686552:Zijun', '6921384:Herman Lee', '2427042:kevin fujiang', '4217323:Dylan Eeles', '1702832:yuanshibuluo', '4044288:Andyman1508', '6976726:denkepeng', '4898322:4710s'}

4.1 完善街道

In [18]:
```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re
import pprint

OSM_FILE_NAME = "map.xml"
```

In [19]:
```
#匹配街道名称最后独立字符串（称谓）
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road",
            "Trail", "Parkway", "Commons"]

# UPDATE THIS VARIABLE
mapping = { "St": "Street",
            "St.": "Street",
            "Rd": "Road",
            "Rd.": "Road",
            "Ave": "Avenue"
            }
```

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


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


def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)

    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'])

    osm_file.close()
    return street_types


def update_name(name, mapping):
    for from_name, to_name in mapping.iteritems():
        #print '(.*)(' + from_name + ')$'
        name_regex = re.compile(r'(.*)(' + from_name + ')$', re.IGNORECASE)
        match_result = re.search(name_regex, name)

        if match_result:
            name = match_result.group(1) + to_name
            break;


    return name
```

4.2 校验tags的value值

In [1]:
```
import re

#用于替换value字段内的奇怪字符
VALUE_REPLACE_CHARS = '[]'
def reject_bad_chars_of_value(src_string, badchars=VALUE_REPLACE_CHARS):
    result = re.sub(badchars, "", src_string)
    return result
#print(reject_bad_chars("邮筒"))

#用于去除错误邮编
REGULAR_POSTCODE =re.compile(r'^[0-9]{6}$')
def is_postcode(src_postcode, regular_postcode=REGULAR_POSTCODE):
    postcode = re.search(regular_postcode, src_postcode)
    if postcode:
        return True
    else:
        return False
#print(is_postcode('12345 '))

#用于规范电话的格式
#如果校验通过，返回正确格式的电话，否则返回空字符串
REGULER_PHONE = re.compile(r'^(86)?(021|21)?([0-9]{8}|[0-9]{11}|400[0-9]{7}|[0-9]{5})$')
def audit_phone(src_phone, regular_phone=REGULER_PHONE):
    new_phone = re.sub(r'[+ \-()]', '', src_phone)
    phone = re.search(regular_phone, new_phone)
    if phone:
        phone_num = phone.group(3)
        phone_type = len(phone_num)
        #固话
        if phone_type==8:
            return '+86-021-' + phone_num
        #手机
        elif phone_type==11:
            return '+86-' + phone_num
        #400电话
        elif phone_type==10:
            return '+86-' + phone_num
        #全国通用电话
        elif phone_type==5:
            return '+86-' + phone_num
        else:
            return ''
    else:
        return ''
#test_phone = ['4008123123','862122163900','+86 6361 2898','021-63914848, 021-63522222','+86 21 38809988','2164312091','86-21-50559888','+2147483647','+862164712821','+18 13621675140','02162883030','+86-21-5160-7888','+86 138 1609 3747','(021) 3356-3996','021-63779282','+86 (0)21-68778787']
#for phone in test_phone:
#    print(audit_phone(phone))
```

5.生成CSV文件

In [2]:
```
import csv
import codecs
import pprint
import re
import xml.etree.cElementTree as ET

import cerberus

import schema

OSM_PATH = "map.xml"

NODES_PATH = "nodes.csv"
NODE_TAGS_PATH = "nodes_tags.csv"
WAYS_PATH = "ways.csv"
WAY_NODES_PATH = "ways_nodes.csv"
WAY_TAGS_PATH = "ways_tags.csv"
REL_PATH = "relations.csv"
REL_TAGS_PATH = "rel_tags.csv"
REL_MEMBERS_PATH = "rel_members.csv"

NODE_TAG_COLON = re.compile(r'^(.*?):(.*)')
LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')

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


SCHEMA = schema.schema
```

In [21]:

```
#专用于处理value值函数
def value_process(current_tag):
    current_tag['value'] = reject_bad_chars_of_value(current_tag['value'])

    #规范电话格式
    if current_tag['key']=='phone':
        tmp_phone = audit_phone(current_tag['value'])
        if tmp_phone!='':
            current_tag['value'] = tmp_phone
            return current_tag
        else:
            print('错误的电话格式: ' + str(current_tag))
            return None
    #去除错误的邮编
    elif current_tag['key']=='postcode':
        if is_postcode(current_tag['value']):
            return current_tag
        else:
            print('错误的邮编格式' + str(current_tag))
            return None
    #其他情况默认保留
    else:
        return current_tag

#专用于处理tag节点的函数
def tag_process(element, problem_chars, tags):
    for tag in element.iter("tag"):
        key_string = tag.get('k')
        if re.search(problem_chars, key_string):
            continue

        current_tag = {}
        current_tag['id'] = element.get('id')
        #去除value中的坏字符
        current_tag['value'] = tag.get('v')

        #根据k的值确定key与type
        if key_string.find(':')<0:
            current_tag['key'] = key_string
            current_tag['type'] = "regular"
        else:
            m = re.search(NODE_TAG_COLON, key_string)
            current_tag['key'] = m.group(2)
            current_tag['type'] = m.group(1)

        #处理value的值
        new_tag = value_process(current_tag)
        if new_tag != None:
            tags.append(new_tag)

    return tags
```

In [22]:

```
# Make sure the fields order in the csvs matches the column order in the sql table schema
NODE_FIELDS = ['id', 'lat', 'lon', 'user', 'uid', 'version', 'changeset', 'timestamp']
NODE_TAGS_FIELDS = ['id', 'key', 'value', 'type']

WAY_FIELDS = ['id', 'user', 'uid', 'version', 'changeset', 'timestamp']
WAY_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_NODES_FIELDS = ['id', 'node_id', 'position']

REL_FIELDS = ['id', 'user', 'uid', 'version', 'changeset', 'timestamp']
REL_TAGS_FIELDS = ['id', 'key', 'value', 'type']
REL_MEMBER_FIELDS = ['id', 'ref', 'type', 'role', 'position']


"""Clean and shape node or way XML element to Python dict"""
def shape_element(element, node_attr_fields=NODE_FIELDS, way_attr_fields=WAY_FIELDS, rel_attr_fields=REL_FIELDS,
                  problem_chars=PROBLEMCHARS, default_tag_type='regular'):
    node_attribs = {}
    way_attribs = {}
    way_nodes = []
    rel_attribs = {}
    rel_members = []
    tags = []  # Handle secondary tags the same way for both node and way elements

    # 检查 node 节点
    if element.tag == 'node':
        for node_attr_field in node_attr_fields:
            node_attribs[node_attr_field] = element.get(node_attr_field)
        #处理该element节点的tag子节点
        tags = tag_process(element, problem_chars, tags)
        return {'node': node_attribs, 'node_tags': tags}

    #检查 way 节点
    elif element.tag == 'way':
        for way_attr_field in way_attr_fields:
            way_attribs[way_attr_field] = element.get(way_attr_field)
        #处理该element节点的tag子节点
        tags = tag_process(element, problem_chars, tags)
        cnt=0
        for node in element.iter("nd"):
            way_node = {}
            way_node['id'] = element.get('id')
            way_node['node_id'] = node.get('ref')
            way_node['position'] = cnt
            cnt+=1
            way_nodes.append(way_node)
        return {'way': way_attribs, 'way_nodes': way_nodes, 'way_tags': tags}

    #检查 relation 节点
    elif element.tag == 'relation':
        #relation节点的属性
        for rel_attr_field in rel_attr_fields:
            rel_attribs[rel_attr_field] = element.get(rel_attr_field)
        #处理该element节点的tag子节点
        tags = tag_process(element, problem_chars, tags)
        #迭代member子节点
        cnt=0
        for member in element.iter('member'):
            member_node = {}
            member_node['id'] = element.get('id')
            member_node['type'] = member.get('type')
            member_node['ref'] = member.get('ref')
            member_node['role'] = member.get('role')
            member_node['position'] = cnt
            cnt+=1
            rel_members.append(member_node)
        return {'relation': rel_attribs, 'rel_tags': tags, 'rel_members': rel_members}
```

In [23]:
```
# ================================================ #
#               Helper Functions                   #
# ================================================ #
def get_element(osm_file, tags=('node', 'way', 'relation')):
    """Yield element if it is the right type of tag"""

    context = ET.iterparse(osm_file, events=('start', 'end'))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()


def validate_element(element, validator, schema=SCHEMA):
    """Raise ValidationError if element does not match schema"""
    if validator.validate(element, schema) is not True:
        #field, errors = next(validator.errors.items())
        #message_string = "\nElement of type '{0}' has the following errors:\n{1}"
        #error_string = pprint.pformat(errors)
        #raise Exception(message_string.format(field, error_string))
        raise Exception(validator._errors)


class UnicodeDictWriter(csv.DictWriter, object):
    """Extend csv.DictWriter to handle Unicode input"""

    def writerow(self, row):
        super(UnicodeDictWriter, self).writerow({
            #k: (v.encode('utf-8') if isinstance(v, str) else v) for k, v in row.items()
            k: (v) for k, v in row.items()
        })

    def writerows(self, rows):
        for row in rows:
            self.writerow(row)
```

In [24]:
```
# ================================================== #
#               Main Function                        #
# ================================================== #
def process_map(file_in, validate):
    """Iteratively process each XML element and write to csv(s)"""

    with codecs.open(NODES_PATH, 'w', encoding='utf-8') as nodes_file, \
         codecs.open(NODE_TAGS_PATH, 'w', encoding='utf-8') as nodes_tags_file, \
         codecs.open(WAYS_PATH, 'w', encoding='utf-8') as ways_file, \
         codecs.open(WAY_NODES_PATH, 'w', encoding='utf-8') as way_nodes_file, \
         codecs.open(WAY_TAGS_PATH, 'w', encoding='utf-8') as way_tags_file, \
         codecs.open(REL_PATH, 'w', encoding='utf-8') as relations_file, \
         codecs.open(REL_TAGS_PATH, 'w', encoding='utf-8') as rel_tags_file, \
         codecs.open(REL_MEMBERS_PATH, 'w', encoding='utf-8') as rel_members_file:

        nodes_writer = UnicodeDictWriter(nodes_file, NODE_FIELDS)
        node_tags_writer = UnicodeDictWriter(nodes_tags_file, NODE_TAGS_FIELDS)

        ways_writer = UnicodeDictWriter(ways_file, WAY_FIELDS)
        way_nodes_writer = UnicodeDictWriter(way_nodes_file, WAY_NODES_FIELDS)
        way_tags_writer = UnicodeDictWriter(way_tags_file, WAY_TAGS_FIELDS)

        relations_writer = UnicodeDictWriter(relations_file, REL_FIELDS)
        rel_tags_writer = UnicodeDictWriter(rel_tags_file, REL_TAGS_FIELDS)
        rel_members_writer = UnicodeDictWriter(rel_members_file, REL_MEMBER_FIELDS)

        nodes_writer.writeheader()
        node_tags_writer.writeheader()

        ways_writer.writeheader()
        way_nodes_writer.writeheader()
        way_tags_writer.writeheader()

        relations_writer.writeheader()
        rel_tags_writer.writeheader()
        rel_members_writer.writeheader()

        validator = cerberus.Validator()

        for element in get_element(file_in, tags=('node', 'way', 'relation')):
        #for element in get_element(file_in, tags=('node')):
            el = shape_element(element)
            if el:
                if validate is True:
                    validate_element(el, validator)

                if element.tag == 'node':
                    nodes_writer.writerow(el['node'])
                    node_tags_writer.writerows(el['node_tags'])
                elif element.tag == 'way':
                    ways_writer.writerow(el['way'])
                    way_nodes_writer.writerows(el['way_nodes'])
                    way_tags_writer.writerows(el['way_tags'])
                elif element.tag == 'relation':
                    relations_writer.writerow(el['relation'])
                    rel_members_writer.writerows(el['rel_members'])
                    rel_tags_writer.writerows(el['rel_tags'])
```

In [25]:
```
process_map("map.xml", validate=True)
```

```
错误的电话格式：{'id': '477661623', 'value': '021-63914848, 021-63522222', 'key': 'phone', 'type': 'regular'}
错误的电话格式：{'id': '2345419578', 'value': '+18 13621675140', 'key': 'phone', 'type': 'regular'}
错误的电话格式：{'id': '3609090494', 'value': '+86 8621 5118 1222', 'key': 'phone', 'type': 'regular'}
错误的邮编格式{'id': '4364315493', 'value': '2000080', 'key': 'postcode', 'type': 'addr'}
错误的邮编格式{'id': '148014167', 'value': '201315 上海', 'key': 'postcode', 'type': 'addr'}
错误的邮编格式{'id': '148014201', 'value': '201315 上海', 'key': 'postcode', 'type': 'addr'}
错误的电话格式：{'id': '159787864', 'value': '8008103088', 'key': 'phone', 'type': 'regular'}
错误的邮编格式{'id': '293504473', 'value': '2000080', 'key': 'postcode', 'type': 'addr'}
错误的电话格式：{'id': '293862126', 'value': '+86 21 64874095*208', 'key': 'phone', 'type': 'regular'}
错误的邮编格式{'id': '307449542', 'value': '20032', 'key': 'postcode', 'type': 'addr'}
错误的邮编格式{'id': '307455604', 'value': '20032', 'key': 'postcode', 'type': 'addr'}
错误的电话格式：{'id': '376223385', 'value': '0862162838711', 'key': 'phone', 'type': 'regular'}
错误的电话格式：{'id': '392578179', 'value': '(+86)21/52068000', 'key': 'phone', 'type': 'regular'}
```

ERROR SHEET

语句：

super(UnicodeDictWriter, self).writerow({ k: (v.encode('utf-8') if isinstance(v, unicode) else v) for k, v in row.iteritems() })

Error: 'dict' object has no attribute 'iteritems'

原因：pyton3与2的语法区别

解决办法: python3中使用：row.items()

语句：

def writerow(self, row): super(UnicodeDictWriter, self).writerow({ k: (v.encode('utf-8') if isinstance(v, unicode) else v) for k, v in row.items() })

Error: isinstance() arg 2 must be a type or tuple of types

原因：python3语法不同

解决办法：使用 str 代替 unicode

文件打开指定utf-8编码，否则读取带特殊人名（非26字母字符）时会出现gnk编码读取错误