OpenStreetMap Data Wrangling with MongoDB

Author: Karthik Rajasethupathy Map Area: [Shanghai, China
http://www.openstreetmap.org/relation/913067]

Why Shanghai?

I am currently living in China. I am not fluent in mandarin, but I can recognize some basic characters (such as street types), and thought it would be interesting to see what kind of inconsistencies and localization problems might arrise in this data.

Problems Encountered

There are many issues with this dataset. To simplify the task, I focused on the street naming conventions and found five treatable classes of issues:

```
1. Malformed Street Names
        <tag k="addr:street" v="3"/> --> This is not a street name
2. Mispellings and Abbreviations
        <tag k="addr:street" v="Wukang Roaf"/> --> "Wukang Road"
        <tag k="addr:street" v="Haigang Ave."/> --> "Haigang Avenue"
3. Pinyin spelling of Chinese Words
        <tag k="addr:street" v="Nandang Dong Lu"/>  --> "Nadang East Road", Dong =
East and Lu = Road!
        <tag k="addr:street" v="Suzhou Dadao Dong"/> --> "Suzhou Avenue East",
Dadao = Avenue and Dong = East
4. Removing Excess Punctuation
        For consistency, changing numbers from #399 -> 399
5. Street address (addr:street) is sometimes in chinese, sometimes in english, and
sometimes in both languages.
        <tag k="addr:street" v="□□□□"/>
        <tag k="addr:street" v="Weihai Road"/>
        <tag k="addr:street" v="□□□□ (Xianxia West Rd)"/>
        Is this a mistake? If so, which cases are correct?
```

Malformed Streets

```
Here are two (not infrequent) values that provided for the addr:street key.
    "S308"
    "3"
I deal with this by skipping this tag when building the json file the osm file.
```

Mispellings and Abbreviations

```
Similar to project 6, there are many abbreviations and misspelled street names
```

```
(Roaf, Rode, Raod -> Road). It could be possible to do some fuzzy matching, but I
went with the direct search and replace.
For abbreviations, I used the same approach as what we used in project 6, and
simply added a couple of frequently used abbreiviations to the mapping dictionary.
```

Pinyin Spelling of Chinese words

```
This is an interesting case. Often times, the chinese pinyin of a word is written
in the english street name. For example: Yongkang __Lu__ instead of Yongkang
__Road__. On the osm street wiki, i found a list of chinese generics for street
mappings and made a mapping dictionary for these names (see
http://wiki.openstreetmap.org/wiki/WikiProject_China#Generics_in_Chinese).
```

Removing Excess Punctuation

```
No further explanation
```

Street Address

```
I first checked the openstreetmap wiki pages to see what the naming convention is
for areas where localization is an issue. Specifically, for China, I came across
the following document: http://wiki.openstreetmap.org/wiki/Multilingual_names#China
From the above link, we can see that a standard entry is typically like this (I
cannot gaurantee that this is the agreed upon consensus, but most contributors seem
to follow this schema):
    name=<Chinese>
    name:zh=<Chinese>
    name:en=<English>
    name:zh_pinyin=<Chinese pinyin (with tones)>
So, the way I wanted to treat this issue is to do the following.
        <tag k="addr:street" v="□□□□"/> -> Correct, leave it
        <tag k="addr:street" v="Weihai Road"/> -> Translate to Chinese
        <tag k="addr:street" v="Nandang Dong Roaf"/> -> Convert pinyin (Dong->
East), Fix spellings (Roaf -> Road), translate to Chinese
        <tag k="addr:street" v="□□□□ (Xianxia West Rd)"/> -> Keep chinese, remove
english
```

Approach

```
Given:
    1. An addr:street value v (the base value, addr:street, not addr:street:*)
    2. A language translation tool (I made an account with google translate api)

Algorithm:
    1 If string contains chinese characters
        1.1 remove all english characters from string -> return result.
    2 Else
        2.1 Correct spellings (Roaf - > Road)
        2.2 Fix abbreviations (Ave. -> Avenue)
```

```
        2.3 Map pinyin spellings to english  (Dong -> East)
        2.4 Remove excess punctuation from String (#3999 Something Road -> 3999
Something Road)
        2.5 Translate processed english string to chinese -> return result
```

Explanation

```
So, I want to first check if there are any chinese (going with a broad definition
here: non-roman) characters in the string. If so, I'm going to assume that the
chinese characters do fully represent the street address. Then, once again, I clean
the string of the "english stuff". This assumption satisfies 99% of mixed language
cases in this dataset. But, I do acknowledge that it is not a foolproof approach to
dealing with mixed strings.
In case 2, we have a string that contains no chinese characters. So, assume it only
contains english. Here, I fix spellings, fix abbreviations, map pinyin to english,
and remove some excess punctuation, then translate the processed string through the
google translate api. The result will replace the original string. Once again, this
is not foolproof if the pinyin for the streetname is wrong.
If fact, there are 2 cases that this approach fails to address:

    1. No spaces between psuedo - pinyin:
        [Pinyin should have the tone information, most pinyin i see in this dataset
does not contain tonal info, hence, i call this psuedo-pinyin.]
        Sometimes, pinyin has spaces, and sometimes the pinyin has no spaces
between characters, e.g.: nihao vs. ni hao
        Google translate api doesn't always deal with no space pinyin (NSP) well.
For example:
        A.
            Actual: ZhongShangNanEr Road -> Google Translate Api -> ZhongShangNanEr
□
            Ideally: ZhongShangNanEr Road -> Google Translate Api -> □□□□□
            What we need: ZhongShangNanEr Road -?>  Zhong Shang Nan Er Road ->
Google Translate Api -> □□□□□
        B. However, in some other cases, Google takes care of this.
            Actual: BaiZhang East Road -> Google Translate Api -> □□□□

        Case A is not handled correctly. Case B is handled correctly. I am not sure
how to address case A, and it is not clear what the difference is between case A
and B. But, I acknowledge it's an issue and we may need a library that can break up
pinyin in an appropriate way.

    2. Mispelled pinyin:
        Yonkand Road -> Google Translate Api -> Yonkand□
        Now, I know the contributor actually meant to write Yongkang Road, instead
of Yongkand Road. These cases need to be addressed on a case by case basis.
```

Data Overview

```
> File Sizes:
    shanghai_china.osm - 458.4 mb
    shanghai_china.osm.json - 527.8 mb
```

```
##Number of Documents
> db.shanghai_osm.count()
2422310

##Number of Nodes
> db.shanghai_osm.find({type:"node"}).count()
2166547

##Number of Ways
> db.shanghai_osm.find({type:"way"}).count()
255451

##Number of contributing users
> var docs =
db.shanghai_osm.aggregate([{"$match":{"type":{"$exists":1}}},{"$group":{_id:"$creat
ed.user",total:{"$sum":1}}},{"$sort":{"total":1}}])
> docs['result'].length
1387

##Top contributing user as a percentage of total documents
>docs['result'][docs['result'].length-1]['total']/db.shanghai_osm.count().100.0
8.272 [%]

##Top 10% of contributing users as a percentage of total documents
>var sum = 0;
> for (var i = Math.round(.9*docs['result'].length);i<docs['result'].length;i++) {
...     sum += docs['result'][i]['total']
... }
> 100.0*sum/db.shanghai_osm.count()
94.4 [%]
```

So, the top 10% of users account for nearly 95 % of all the contributions.

Additional Ideas

Reliability Scores

I think it would be great if each tag in the osm file has an associated reliability score. The motivation behind this is to assure other correctors they can ignore a particular data and focus on auditing or cleaning up other other data. I think the reliability score should be a function of the # of contributors that have "checked" or "verified" a record, weighted by their influence (this is a contributor score based on how many verified rows they've contributed to OSM). This could allow a potential contributor to direct their auditing focus to specific rows in the data that are under-verified.

Implementation -- Changes to schema

So, let's quickly review the osm xml structure: nodes, ways, and relations. Within each of these primitives, users can assign extra tags. For example, a node is just a point. But, an additional tag can used to mark this point as a standalone object such as a stoplight or a well. Now, to implement

a reliability system, I would advocate for the addition of a couple more tags to each node (way, and relation). For example:

Before:

```
<node id="1" version="1" changeset="1" lat="54.045134" lon="12.2539381"
user="Karthik" uid="1" visible="true" timestamp="2015-12-26T09:43:19Z">
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
 </node>
```

After:

```
<node id="1" version="1" changeset="1" lat="54.045134" lon="12.2539381"
user="Karthik" uid="1" visible="true" timestamp="2015-12-26T09:43:19Z" >
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
  <checkins users=3 last_cid=3 score=>
    <tag cid = 1 user="Karthik" timestamp="2015-12-26T23:31:00Z"
reliability_score_at_checkin="3"/>
    <tag cid = 2 user="Jimbo" timestamp="2015-12-28T23:31:00Z"
reliability_score_at_checkin="0"/>
    <tag cid = 3 user="Ben" timestamp="2015-12-29T23:31:00Z"
reliability_score_at_checkin="16"/>
  </checkins>
 </node>
```

In the xml entry above (After), I have added a primitive within the node element called "checkins". This section will capture any and all information about who/how/when contributors validated this entry. The checkins header contains some summary information about how many users have verified this point, and the cid (checkin id) of the most recent verification, as well as a score value which is the reliability score for this entry. This score would be a function of the sub tags, and I would propose using something like a sigmoid function of the reliability scores of the contributors to calculate this.

What about modifying an existing entry?

Even with these changes, and assuming that we have the appropriate functions in place to calculate reliability scores, I can see a whole host questions that we would need to address in order to implement this correctly. For example, let's assume we're looking to verify the entry above. We happen to be at the lat/lon specified above and there is in fact NO city limits traffic sign. Maybe the city took it down. How should we record such a change? Do we create a new entry and mark this one as void or invalid? Or, should we modify this entry? And, if we can modify this entry, what happens if our reliability score is low (maybe we are a new user)? Do we need to wait until we get some sort quorom for this entry change? We need a framework that can manage updates and accept changes in an automated way when enough of the community can agree upon it.

Reliability for nodes versus ways and relations

OSM does a great job of defining complex structures from building blocks. - Ways are made up of (between 2 and 2000) nodes are typically linear separators. - Relations can be composed of two or more members (nodes, ways and/or other relations) and can represent a multi-polygon area or a bus routes, etc

Assuming we can verify single node entries, assume there is a way as follows:

```
<way id="26659127" user="Masch" uid="55988" visible="true" version="5"
changeset="4142606" timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
```

This way has 3 node references. If each node reference is independently verified, should that influence the reliability of this way? Or, conversely, if a contributor were to verify this way, should that influence the reliability of its sub nodes? I think that indeed both dependencies should be taken into account. But, these are things that need to be thought through and weighted appropriately. And, as we think about how transitive reliability is, we need to consider the extension to relations as well, which are the most complex primitive that osm allows.

Summary - Reliability

I proposed a very simple implementation of reliability scores. While verifying an existing osm element seems to be simple enough. It is not clear how we should deal with "updates" or "modifications" to existing entries. Intuitively, a reliable modification, backed by a large number of contributors should be "accepted" after a certain quorom, but clearly a framework must be built to support this.

Furthermore, there is a question of how reliable is an entry that is composed of other reliable entries; that is, the transitivity of reliability scores. This is an important issue that needs thought through because osm primitives are building blocks that can be used to construct more complex entries.

More Data Exploration With MongoDB

Contributions by year

```
> db.shanghai_osm.aggregate([
    { "$project": {
         "xy": { "$substr": [ "$created.timestamp", 0, 4 ]}
         }},
    { "$group": {
```

```
            "_id": "$xy",
            "n": {$sum: 1}
        }}]);

Year: Contributions
2007: 83634
2008: 19634
2009: 39915
2010: 85846
2011: 178272
2012: 377958
2013: 345522
2014: 559830
2015: 731699
After a dip from '07 to '08 and a minor recovery in '09, contributions nearly
doubled from '10 to '11 and '11 to '12. There was again a slight dip from '12 to
'13, but a nice recovery in the last few years.
```

Bakeries and Massage Parlors, and where are the KTV's?

Anyone that's every lived shanghai knows that its full of delicious bakeries and lined with massage parlors. I would wager a conversative guess that for each mall or big shopping center, there are 10 bakeries and 5 massage parlors. I wanted to see if osm data is consistent with these expected ratios.

```
> db.shanghai_osm.aggregate([{"$match":{"shop":{"$exists":1}}},
{"$group":{"_id":"$shop","count":{"$sum":1}}}])
>
{
"result" : [
    ...
    {
        "_id" : "massage",
        "count" : 1
    },
    {
        "_id" : "bakery",
        "count" : 40
    },
    {
        "_id" : "mall",
        "count" : 139
    },
    {
        "_id" : "supermarket",
        "count" : 406
    }
    ...
],
"ok" : 1
```

}

Let's first assume that every supermarket has a bakery. This is a pretty safe assumption. Then, we have 40 bakeries and 406 supermarkets so the total number of bakeries is 446. There are 140 malls, and only 1 massage parlor!. The raio of malls:bakeries:massage parlors is 1:3:.002. I would the proportion of malls:bakeries is well reflected if my assumption isn't too far off. However, massage parlors are severely underdocumented (~3 orders of magnitued too low). Now, it could be that a different tag, such as amenity, contains this data. After all, there are more amenities than shops in the shanghai osm data:

```
> db.shanghai_osm.find({amenity:{"$exists":1}}).count()
8329
> db.shanghai_osm.find({shop:{"$exists":1}}).count()
1388
```

I suspect there's quite a bit of overlap between the two tags as well. Under amenities, there are 179 cafes, 220 fast food, and 770 restaurants. I could see bakeries falling under each of these umbrellas. For massage parlors, the only possible namespace under amenities that it might belong to is spa. However, there is only 1 entry here.

I would also like to note that KTV (karoke) shops are also severly underrepresented. In amenities, there are only 3 records (under pub:ktv), and there are 0 records for this under shop (no ktv, entertainment, etc).

Conclusion

This has been a challenging and fun assignment. The shanghai osm data is obviously incomplete, but its encouraging to see contributors go up on a yearly basis. Another metric that could indicate if the contributing community is growing is unique contributors per year. If this number is growing, I believe it is an even greater indication that the support for Shanghai OSM data is growing. I focused my work on auditing the addr:street tags. But, I can imagine the difficulties in generalizing these approaches to other tags. It made me think about what challenges exist in building a generalized auditing framework. Ultimately, it is probable that that each tag must be treated in its own way. But, at a bird's eye view it seems like we are applying the same general approach to auditing each tag. So, although this process feels quite specific during implementation, there is some general methodology that is being followed. It was reassuring to experience that.