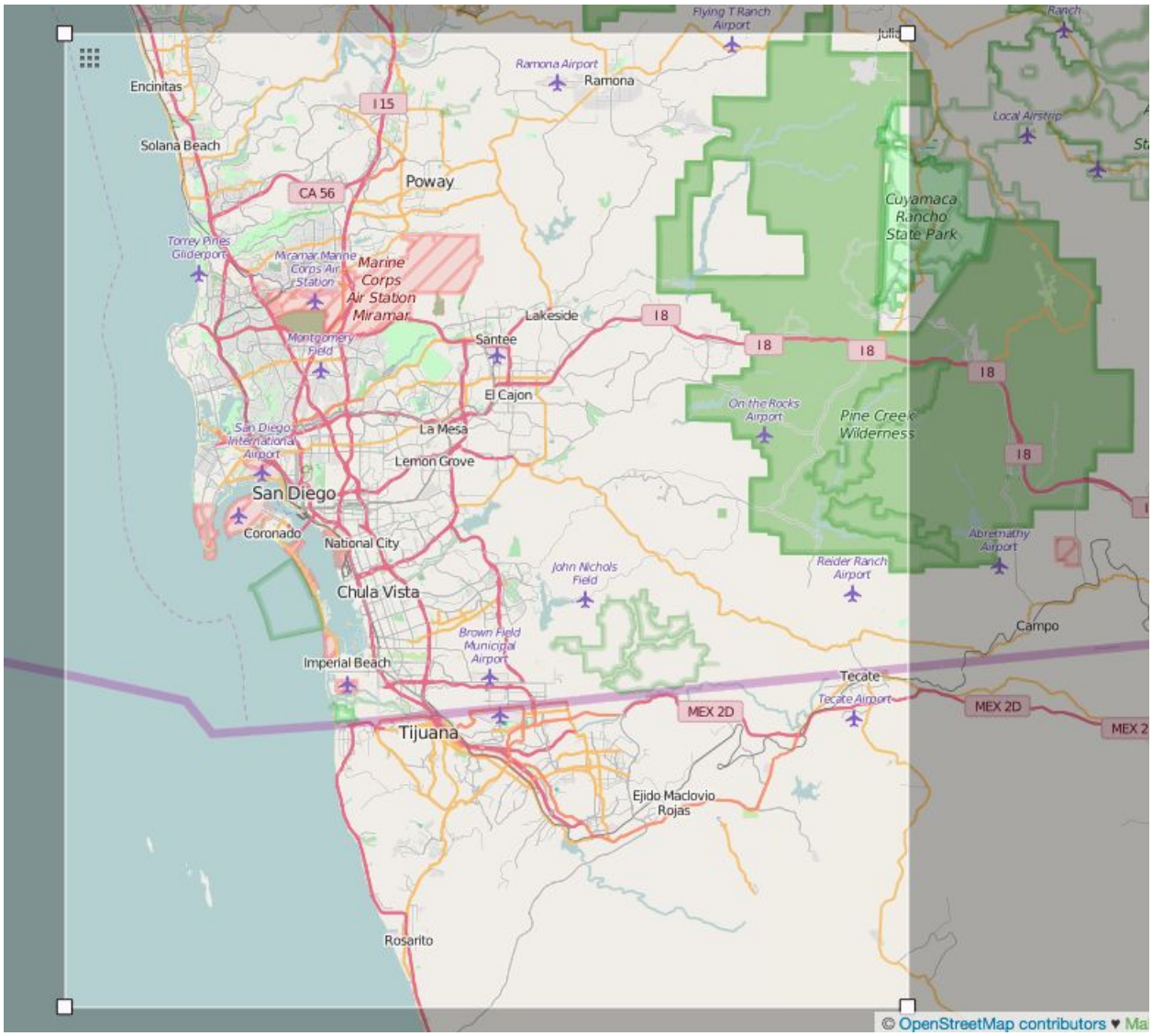
OpenStreetMap Project
Data Wrangling with MongoDB

Brian Chase

Map Area: San Diego, CA, USA - Tijuana, Mexico Metro Area



http://metro.teczno.com/#san-diego-tijuana

1. Problems encountered in map

Over-abbreviated street names

The first thing I ran was project4.py audit code which I used to print out a file of street names grouped by suffixes (example: Ave). Here I could print out the ones that did not match my expected suffixes.

The problem is that the street suffixes are not the only things abbreviated. Abbreviated words could be anywhere in the string—but mostly at the beginning and end of the street name

An example
E Washington Av -> East Washington Ave

Using project4.py process_map(), I modified the street name endings and modified the street name beginnings (for spanish) while shaping the data into the JSON format.

Street names have different conventions due to the map containing Mexico and USA

This issue is an extension of the over-abbreviated street names. Part of the map is in the US and the other part is in Mexico. This was a challenge for me just because I do not speak spanish and I am not familiar with the abbreviations.

To figure out the abbreviations, I used the project4.py audit code to create a file with street names grouped by the first word—since spanish is the opposite of english and has the name of the street after the spanish word for street (calle)—for example: calle primera.

How I handled this, I used the file created by project4.py audit code to find all the english abbreviations, fix them in the file and then the same for the spanish streets. In every iteration, it would skip printing ones that matched the expected endings and beginnings so that there would be a shrinking list to look through.

Another issue is that while spanish streets had a certain format, some streets in San Diego, CA had spanish words but english format—an example is “Vista Camino” or “Old el camino real”. If I were to spend more time on this, I would write the find and replace functionality to find any whole words (separated by spaces) instead of just the first word or the last word.

Postal codes and cities are inconsistent format

UPDATE (11/7/2016): *Added an auditing function for postal codes (function audit_postcodes). The regular expression that I created showed that there are a few places that have different formats---and only one true mistake (zip code 921009). I've cleaned programmatically the different variants of the zipcode, but I have not fixed the one erroneous zip code. I could fix that one entry in the data by hand.*
UPDATE (11/10/2016): *Added code to clean postal code in shape_element().*

After some cleaning, I used datawrangle_project5_xml2json.py to port the .osm file to .json format. And used mongoimport to import the data into mongoDB.

Doing a database search for postal codes

```
client.examples.map.aggregate([{"$match":{"address.postcode":{"$exists":1}}},
        {"$group":{"_id":"$address.postcode", "count":{"$sum":1}}},
        {"$sort":{"count":1}}] )
```

and cross referencing them against valid zip codes found from the sdcourt.ca.gov website: http://www.sdcourt.ca.gov/pls/portal/docs/PAGE/SDCOURT/GENERALINFORMATION/FORMS/ADMINFORMS/ADM254.PDF

I get a list consisting of the following cases:

1. Mexican postal codes that start with 22 and was not in my San Diego, CA, USA list. I can get a list of valid postal codes from a website, but it may be difficult to make sure if they fall in the boundary of the map. (Example: 22440)
2. Full 9-digit US postal codes. (Example: 92102-4810)
3. Postal code starting with 'CA'. (Example: CA 92109)
4. Mistakes (Examples: 'San Diego, CA', 'Scripps Ranch Blvd', '9839')
5. Postal codes not in the region boundary—may be (Example: '92016')

Cities data

The cities data is similar to the postal codes—it is easy to spot since they are the ones that have a low occurrence.

Examples: "tijuana" (lack of caps), "Bonita" (not a city), "Santee, CA" (adding california abbreviation).

2. Overview of the data

Student provides a statistical overview about their chosen dataset, like:

- size of the file
- number of unique users
- number of nodes and ways
- number of chosen type of nodes, like cafes, shops etc

File Sizes

san-diego-tijuana_mexico.osm …… 540.7 MB

san-diego-tijuana_mexico.osm.json …… 603.3 MB

Number of documents

```
> client.examples.map.find().count()
1641520
```

Number of nodes

```
> print client.examples.map.find({"type":"node"}).count()
1522372
```

Number of ways

```
> print client.examples.map.find({"type":"way"}).count()
119028
```

Number of unique users

```
> len(client.examples.map.distinct("created.user"))
1135
```

3. Other ideas about the dataset

User entry data cleaning

An idea to improve the data is to have warnings to the user of flags marking questionable content in individual documents when an entry is “out of the norm” for that field.

For example, if someone types in ‘tijuana’, the website could suggest to the user to replace it with ‘Tijuana’ since that has many other similar entries and tijuana has none. If the user continues anyway, there could be a tag that flags it as needing to be double checked.

Other databases (for example, one for zipcodes based on areas) could be cross-referenced to create the suggestions. It can also be done in the background and questionable content that the cleaning algorithms finds can be sent to incentivized or otherwise motivated people to verify if the data is correct.

How might users be motivated to verify the data?

There could be user profiles where points are given or badges for participation. Another idea is to provide an app framework/API for mobile apps where people can use the data for different uses and people can win prizes for keeping their locations up-to-date.

Anticipated issues with proposed solutions

If the data entry is forced to be within certain strict guidelines, it could impact the flexibility and accuracy of the map. Maps are constantly changing and the data may not be able to keep up-to-date when a zip code is added or a field needs to be changed in format. Even if there was a lot of resources to keep this strict guidelines up-to-date, it is not an efficient means to keep an open map with few official site moderators. That is why it is crucial that the flexibility of the data entry is maintained in that any automated improvement is delivered as a suggestions. Even as a suggestion, it could influence how people enter the data depending on how they interacted with the suggestions and whether the suggestions are the selected default option.

Using outside sources of data for the suggestions could be problematic in that the other sources of data may not be kept up-to-date or have other accuracy problems that you most likely have control over. Having a suggestion reporting flag could help flag problems with the suggestions if the users were savy enough. And if there were problems with the outside sources, that could force making a copy of the outside sources and then the centrally managed data issue arises again.

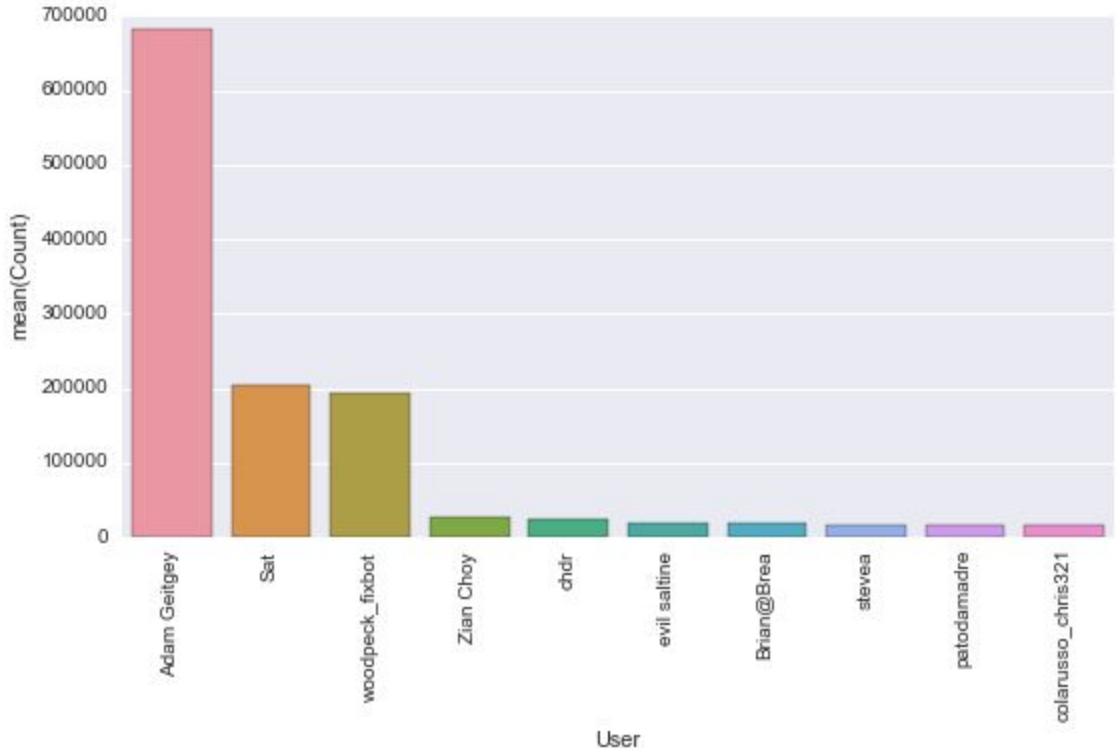
Issues incentivizing or other means of motivating users for participation

Users could make changes for changes sake and---since it would be difficult to verify if the changes are correct---incorrect data would decrease the quality of the data. An idea to maintain the quality of the data is to only change the data after several people report the data. It'd have to be open about how many reports would need to be made to change the data or else people would be dissuaded from trying.

The creation and maintenance of an API would need human and other resources. It'd need to have a viable business model.

Additional data exploration using MongoDB queries

Top 10 contributing user




```
>client.examples.map.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}])
{u'_id': u'Adam Geitgey', u'count': 681384}
{u'_id': u'Sat', u'count': 203082}
{u'_id': u'woodpeck_fixbot', u'count': 193496}
{u'_id': u'Zian Choy', u'count': 26938}
{u'_id': u'chdr', u'count': 24300}
{u'_id': u'evil saltine', u'count': 19682}
{u'_id': u'Brian@Brea', u'count': 18463}
{u'_id': u'stevea', u'count': 15916}
{u'_id': u'patodamadre', u'count': 15164}
{u'_id': u'colarusso_chris321', u'count': 15032}
```

Number of users appearing only once (having 1 post)

```
>client.examples.map.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {"$group":{"_id":"$count",
"num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":1}])
[ {u'_id': 1, u'num_users': 203} ]
# “_id” represents postcount
```

Top 10 Biggest religions

```
pipeline = [{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
        {"$group":{"_id":"$religion", "count":{"$sum":1}}},
        {"$sort":{"count":-1}}, {"$limit":10}]
```

{u'_id': u'christian', u'count': 972}
{u'_id': None, u'count': 46}
{u'_id': u'jewish', u'count': 9}
{u'_id': u'muslim', u'count': 5}
{u'_id': u'hindu', u'count': 4}
{u'_id': u'buddhist', u'count': 4}
{u'_id': u'ascended_master_teachings', u'count': 1}
{u'_id': u'bahai', u'count': 1}
{u'_id': u'scientologist', u'count': 1}
{u'_id': u'sikh', u'count': 1}

Top 10 amenities

```
pipeline = [{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":10}]
```

{u'_id': u'place_of_worship', u'count': 1046}
{u'_id': u'fast_food', u'count': 695}
{u'_id': u'restaurant', u'count': 583}
{u'_id': u'school', u'count': 405}
{u'_id': u'bar', u'count': 291}
{u'_id': u'cafe', u'count': 218}
{u'_id': u'fuel', u'count': 215}
{u'_id': u'bank', u'count': 148}
{u'_id': u'toilets', u'count': 101}
{u'_id': u'library', u'count': 92}

Most popular cuisines

```
pipeline = [{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}},
{"$group":{"_id":"$cuisine", "count":{"$sum":1}}},{"$sort":{"count":-1}}, {"$limit":10}]
```

```
{u'_id': None, u'count': 227}
{u'_id': u'mexican', u'count': 50}
{u'_id': u'pizza', u'count': 41}
{u'_id': u'italian', u'count': 27}
{u'_id': u'american', u'count': 25}
{u'_id': u'sushi', u'count': 23}
{u'_id': u'chinese', u'count': 23}
{u'_id': u'thai', u'count': 18}
{u'_id': u'burger', u'count': 17}
{u'_id': u'japanese', u'count': 16}
```

Conclusion

In cleaning and exploring this dataset, it is clear there are many errors from users and perhaps many more cases that I have not uncovered. I have in this project covered some of the more simple cases.