OpenStreetMap Data Case Study

Area covered:

- **Scottdale, AZ and outlying Phoenix and suburbs area** : https://www.google.com/maps/place/Scottsdale,+AZ/@33.6738949,-112.1386422,10z/data=!3m1!4b1!4m5!3m4!1s111.9260519 (https://www.google.com/maps/place/Scottsdale,+AZ/@33.6738949,-112.1386422,10z/data=!3m1!4b1!4m5!3m4!1111.9260519)
- Cities represented: Phoenix, Scottsdale, Paradise Valley, Fountain Hills, Tempe
- Area grid from 32.80 to 34.07 degrees lattitude and from -112.83 to -111.22 degrees longitude
- Area chosen to include my neighborhood plus surrounding neighborhoods to give large enough area for project

Challenges encountered in the OSM mapping data

With the OSM data, initially looking at the street address and cuisine data, we found a number of problems, only some of which could be addressed programmatically unless unilateral decisions were made that would affect other OSM mappers.

- **Many** addresses were using an invalid format for street name by appending an apartment or unit number, which should be in another field/tag such as *flat* or *unit*. Example: *W Bell Rd #E108*

- Spanish names for streets did not allow for straitforward regular expressions to capture and clean street names, i.e. *Avenida del Yaqui.*

- For cuisine values, no standardization among possible choices, and no obviously better choice. Example: *burger;american* and *american;burger* both are used about as often. Also, both *bbq* and *barbecue* are used, but they mean exactly the same thing.

- Invalid types of things included in cuisine, such as *vegan* (which should be included in *diet* tags), or *drive-through*, which has its own tag to indicate this.

- Multiple types of incorrect formatting: prepending an underscore, using spaces, using commas instead of semicolons, and of course, capitalization when only lowercase should be used.

Appended apartment/unit numbers

Problem description
Many of the street name tags had apartment or unit numbers appended at the end (i.e. *W Bell Rd #E108*). However, this is incorrect format; an apartment or unit number is not part of the street name. As can be seen in the OSM online documentation and help forums (see the seperate document for online references used), this should be indicated with the *addr:unit*, *addr:flats* or possibly the *addr:housenumber* tags.
Likely cause
This may very well be a result of automated parsing of Google Map addresses or some other shemes. For example, a Google search result has "7366 E Shea Blvd #111 Scottsdale, AZ 85260". It is easy to see a regular expression would parse this and put 7633 as the house number, and "E Shea Blvd #111" as the street, which works as expected when you have an address like "123 E Main St" without a unit number.
One possible solution:
There have been multiple proposals to deal with the various possibilities for the finer grained numberings such as unit, house or apartment number, but no consensus (see the online references listed). The idea favored here is to use the addr:unit tags, as "unit" is suitably generic to handle the different situations. Using *addr:flat* might make the most sense for apartments, but a *"flat"* is an idiomatic expression perhaps used more in Europe or the UK. It would make sense to extend this scheme with an *addr:unit2* and possibly *addr:unit3* to address situations when there are finer divisions to be made, for example in a unit that is inside a store that is inside a shopping mall which has an address with a street number.
One possible implementation
When parsing a street name and finding a number at the end after "Ste", "Suite", "Room", "Rm" or "#", then take the number, strip out the *"Ste"* (or *"Room"* etc), and put it into an additional *addr:unit* tag. Use the content besides the "Ste" and the corresponding number as the street name in the *addr:street* tag. Both the *addr:street* and the *addr:unit* subtags are to belong to same list of way or unit (sub)tags.
(Note that a similar solution was implemented to add additional subtags for diet and drive_through to supplement the cuisine tags. See section *"Invalid things included in cuisine value"*.)
Possible consequences of this proposal and action taken
Doing this would unilaterally rewrite a great number of entries made by OSM map developers, and it would be in the absense of any consensus for the proper taggings to use for these situations. *Therefore, nothing was done.*

Spanish street names not conforming to English patterns

Problem description
The regular expressions used to process the street type (i.e. "Street", "Avenue", "Road", etc) at the end will not work for streets with Spanish names. In the normal case, we would parse "123 Main St" and check the ending ("St") to see if it should cleaned (to "Street"). But with, say, "9405 S Avenida del Yaqui", the "Avenida" (Spanish for "Avenue") is not at the end, so this would not work.
Likely cause
No mystery here; the regular expression used here was not designed for Spanish addresses.
Possible solution
There would be no straitforward and fool-proof way of doing this, as regular expressions often miss some edge cases. One possibility would be to check for Spanish words surrounded by spaces such "de", "del" "de la", etc, and then search for "Av", "Avenida", "Cll", "Calle", "Cir", "Circulo", etc, in order to see if these need to be cleaned.
Possible consequences of this proposal and action taken
As mentioned, it is not easy to find a really good regular expression that will capture all possible addresses including corner cases. Also, it is not obvious what should be done if we see "9405 S Av del Yaqui". Should the "Av" be converted to Avenida or to Avenue? This is ambiguous. Moreover, we sometimes see "North" and "Norte" in these Spanish addresses, so there is no way to know in advance if Spanish or English words should be used in the cleaned address.
It turns out that there were very, very few cases of an abbreviated "Av" or "Cll" in the context of Spanish street names, so this turns out not be a big issue. *As a result, no action was taken.* However, it could be the case in other cities where this is a nontrivial problem.

Cuisine values: lack of standardization

Problem description
There are multiple ways that the same cuisine type is described, with no apparent standardization to serve as a guideline. For example, we see both *"bbq"* and *"barbeque"*; and we see both *"american;hamburgers"* and *"hamburgers;american"*
Likely cause
There is no standard that is spelled out in the OSM wiki, and it is difficult to imagine a standard that would spell out which word or phrase to use for each and every case. Moreover, there is nothing to stop to OSM mapper from using their own terms.
Possible solution
The only solution found was just to go by the convention of the most used word/phrase already in place, with outlier words or descriptions replaced by the most popular words. For example, both *coffee* and *coffee_shop* were used, but *coffee_shop* was used much more often. So the data cleaning for the cuisine values used a dictionary to map some of the less used terms to the more standard terms which would replace them.
Also, when there were obvious typos (*Roast_Beel* instead of *roast_beef*) they were corrected when possible, and when the correct subsitute could not be inferred (i.e. a cuisine value of *ret*) then the cuisine value was ignored. If a cuisine value was to be ignored, if there were other values used (i.e. if we saw *mexican;ret*) these other values were retained, but if there were no other values used, then we ignored the cuisine tag.
Possible consequences of this proposal and action taken
This process clearly identifies some outliers that would be best eliminated in the face of more widely used alternatives. But by relying on convention we inevibilly hit some situations in which one use was about as popular as the alternative. In these cases, we did not make any replacements.
Despite the cases in which there was no clearly obvious advantage to relying on the most popular usage, we went with this guideline and implemented it in the actual clean up because it was at least a straitforward solution, and there were some clear cases in which outliers should be eliminated.

Cuisine values: invalid things included

Problem description
Some entries for cuisine value had invalid included as part of cuisine. For example, being vegan is considered as applying to diet, not cuisine, and belongs in a value for a *diet* tag. Also, being a drive-through is not part of the cuisine and should be indicated instead in a *drive_through* tag.
Likely cause
As mentioned earlier, there are no quality control or enforcement mechanisms to stop a OSM mapper from putting in whatever they think is appropriate.
Possible solution
When we see vegan or vegetarian in the cuisine value(s), we remove it from there and put it in a field for the tag for *diet if there is not a tag for diet already there*. Similarly, when we see an indication of drive-through in the cuisine values, we remove it from there and create a *drive_through* field (for where a *drive_though* tag would be) in the dictionary, but only if there is not already a *drive_through* tag already existing. These tags (*diet* or *drive_through*) as well as the *cuisine* tag would be subtags to the the same *way* or *node* parent tag.
Possible consequences of this proposal and action taken
This time there did not appear to be any obvious down-side to doing this. There could be some confusion caused by making a correction whose motivation is not understood, but this can always happen if/when corrections are made. *As a result, these changes were made when cleaning the data.*

Cuisine values: invalid formatting

Problem description
There are multiple instances of incorrect formatting of the cuisine values. Examples include using a comma instead of a semicolon as a seperator, using spaces between cuisine values, prepending underscores before a cuisine value, and capitalizing the cuisine value.
Examples:
Middle Eastern
pizza,_burgers,_sandwiches,_salads,
Likely cause
It would seem some OSM mappers were not paying attention to the common norms for formatting.
Possible solution
Eliminating the prepended underscore, converting all cuisine values to lowercase, changing a space between words within a single cuisine value to an underscore (i.e. middle eastern converted to middle_eastern), and removing spaces between successive cuisine values.
Possible consequences of this proposal and action taken
Again this might surprise some of the OSM mappers to have their cuisine values cleaned up, but it seems that it would be worth it in the name of establishing greater standardization. *As a result, we went ahead with these changes during the cuisine value clean up.*

A look at the cuisine data

Note: in order to improved readability and to simplify queries, we created a view *nodes_ways_tags_union* as a union of the subtags under *node* and *way* tags.

```
sqlite> create view nodes_ways_tags_union AS select * from ways_tags UNION ALL
select * from nodes_tags;
```

Most popular cuisines

The first, most obvious question is: What are the most popular cuisines?
*The top answer, **burger**, is no surprise. This being Arizona, **mexican** for 2nd place is not a surprise either.*

```
sqlite> select value, count(*) as count from nodes_ways_tags_union where key
='cuisine'
...> group by value order by count desc limit 10;
burger|356
mexican|190
pizza|134
sandwich|111
coffee_shop|102
american|91
chicken|44
italian|42
chinese|35
asian|19
sqlite>
```

Post codes with most cuisine entries

The next question to ask is: which postal (zip) codes had the most entries for cuisines entered?

```
sqlite> select value, count(*) as count from nodes_ways_tags_union WHERE key
 = 'postcode'
...> and id in (select id from nodes_ways_tags_union where key='cuisine')
...> group by value order by count desc limit 20;
85374|24
85301|16
85201|15
85209|15
85382|15
85258|14
85281|14
85018|13
85120|12
85268|10
85297|10
85305|10
85014|9
85012|8
85260|8
85306|8
85013|6
85202|6
85017|5
85128|5
sqlite>
```

Baseline: Postal codes with most amenities

We should take a baseline for the above question, and see if the postal codes with the most cuisine entries corresponds roughly with the most used postal codes for amenities.
There is a semi-good overlap, but not quite as tight a correlation as I predicted. Of the top 10 postal codes having cuisine entries, just 6 were in the top 10 postal codes having amenities, and of the remaining 4, 3 were in the top 20 for amenity postal codes and 1 was not in the top 20.

```
 sqlite> select value, count(*) as count from nodes_ways_tags_union WHERE ke
y = 'postcode'
...> and id in (select id from nodes_ways_tags_union where key='amenity')
...> group by value order by count desc limit 20;
85301|78
85120|72
85201|54
85209|42
85306|41
85375|41
85305|40
85374|39
85018|33
85128|32
85281|32
85382|32
85142|27
85268|27
85119|26
85303|26
85302|24
85202|23
85003|22
85207|22
sqlite>
```

Post codes with the most Mexican restaurants

Now, for the real reason we looked closer at cuisine values! Which postal codes have the most Mexican restaurants?

```
 sqlite> select value, count(*) as count from nodes_ways_tags_union WHERE ke
y = 'postcode'
...> and id in (select id from nodes_ways_tags_union where key='cuisine' and
 value like '%mexican%')
...> group by value order by count desc limit 20;
85268|4
85018|3
85034|3
85258|3
85012|2
85120|2
85128|2
85201|2
85295|2
85301|2
85306|2
85308|2
85382|2
85004|1
85013|1
85016|1
85021|1
85048|1
85086|1
85119|1
sqlite>
```

Hmm, not as many Mexican restaurants in the top zip codes as expected, but at the same time, not so bad. There are easily 10 or more Mexican restaurants per zip code in the most popular or more frequented areas in Phoenix and Scottsdale. So this looks like we are showing maybe 20% or so of the expected number.
To get a better understanding, we look also at sushi restaurants for a comparison.
And as a result, we get **just 4 sushi restaurants total**! This a serious undercount, much more so than the estimated 20% of actual number of Mexican restaurants showing up.

```
sqlite> select value, count(*) as count from nodes_ways_tags_union WHERE key
 = 'postcode'
...> and id in (select id from nodes_ways_tags_union where key='cuisine' and
 value like '%sushi%')
...> group by value order by count desc;
85282|1
85286|1
85305|1
85382|1
sqlite>
```

Miscellaneous questions and queries; data overview

We now ask some miscellaneous questions pertaining the data in general.

Some stats for the data

Data size

```
phoenix.osm     733.0 MB
OSM_Phoenix.db  424.4 MB
nodes.csv   268.5 MB
nodes_tags.csv  11.9 MB
ways.csv    26.3 MB
ways_nodes.csv  90.8 MB
ways_tags.csv   69.2 MB
```

Number of nodes

```
sqlite> select count(*) from nodes;
3156323
```

Number of ways

```
select count(*) from ways;
426144
```

Number of users contributing to nodes and ways tags

```
sqlite> select count(distinct(user)) from ( select user from ways UNION sele
ct user from nodes);
1517
```

*Number of users contributing to nodes tags but **not** to ways tags*

```
select count(distinct(user)) from nodes WHERE not user in (select distinct(u
ser) from ways);
447
```

*Number of users contributing to ways tags but **not** to nodes tags*

```
select count(distinct(user)) from ways WHERE not user in (select distinct(us
er) from nodes);
149
```

So, based on queries above, it would appear that the vast majority (but not all) of OSM map developers create content for both ways and nodes.

Most used subtags for nodes tags

```
sqlite> select key, count(*) as count from nodes_tags group by key order by
 count desc limit 20;
highway|87707
power|24577
natural|20466
crossing|16928
bicycle|11231
horse|11099
supervised|11022
name|10039
traffic_calming|9901
amenity|7860
state|7092
city|6939
street|6568
housenumber|6477
created_by|6334
barrier|6039
postcode|4585
country|4230
leisure|2915
noexit|2676
sqlite>
```

Most used subtags for ways tags

```
sqlite> select key, count(*) as count from ways_tags group by key order by
count desc limit 20;
highway|279751
name|180313
oneway|137967
lanes|118747
surface|104190
county|91987
name_base|81919
reviewed|80842
cfcc|77187
building|76253
name_type|76022
name_direction_prefix|73695
service|61983
maxspeed|45898
smoothness|37771
city|29878
state|24899
country|22893
source|20572
cycleway|16731
sqlite>
```

Finally, two integrity check type queries

Are any invalid node IDs (node_id) appearing in the node subtags for way tags? That is, do the node_ids for these subtags all refer to an existing node?
No, all the node IDs of the subtags refer to existing nodes.

```
select count(distinct(node_id)) from ways_nodes WHERE NOT node_id IN (select
 id from nodes);
0
```

Are all the IDs for nodes distinct, as expected?
Yes, we see the same number of nodes and distinct node IDs.

```
select count(*) from nodes;
3156323

select count(distinct(id)) from nodes;
3156323
```

Additional ideas and suggestions

It seems that there might be longstanding issues having to deal with the lack of standardization of cuisine types. Since there is no real enforcement mechanism and the OSM developers are mostly working on a voluntary basis, having some rules to force people to use certain conventions would be unrealistic. Perhaps the best way to address this issue is with more information put into the various OSM wiki pages, especially for choices that are not obvious.

For example, it is not apparent that *vegan* and *vegetarian* are not cuisine types, but are rather descriptors for *diet* tags. Likewise, being a *drive_through* is not a cuisine descriptor (although this is more obvious than in the case of vegan/vegetarian). These kinds of things could be highlighted in the main wiki page for cuisine (https://wiki.openstreetmap.org/wiki/cuisine (https://wiki.openstreetmap.org/wiki/cuisine)) under a "What is not a cuisine feature" section or something along those lines.

For the case of the apartment or unit numbers being included in the street name (discussed but with no solution implemented), one obvious recommendation would be to standardize around the *unit* tag, as it is generic enough to cover the cases of apartments as well as other random kinds of finer grained location or address divisions. In addition, you could also have *unit1*, *unit2*, or *unit3* tags, just as we have for *address2* tags, to cover any degree of subdividing that is needed.

But it seems like many of the cases where apartment or unit numbers were included in the street names are likely due to automated parsing of addresses like "123 Main St #B101" which just parses the unit number as part of the street because it is at the end of the string, where the street name usually is. For these cases, it might make sense to run a batch job semi-regularly to capture the unit numbers and generate a unit tag (assuming there was such a standardization).

Conclusion

This project looked at the OSM map data for the Scottsdale and outlying Phoenix suburbs, paying particular attention to the cuisine tags and to a lesser extent the address information.

We found that there was a low degree of standardization with the cuisine types, with different tags like *mexican* or *mexican_food* being used for the same thing. Also we found many formatting issues, such as spaces between cuisine words, use of captital letters, use of commas instead of semicolons, and leading underscores. Finally, we found cases where *vegan*, *vegetarian*, and *drive_through* were used for cuisines, despite they're not being cuisine descriptors. The lack of standardization was cleaned up by relying mostly on the most common usage to substitute the more standard terms for the less standard terms. The formatting was cleaned up by stripping out the things that weren't needed, and in some cases putting in underscores instead of spaces between words. And the *vegan*, *vegetarian*, and *drive_through* cases were handled by eliminating them from the cuisine tag and putting them instead into *diet* or *drive_through* fields.

When looking at the address street names, we found many cases in which the unit or apartment number was included in the street name. No solution was implemented, as we focused on cuisine instead, but we did note that this could be addressed by standardizing on the *unit* tags and then reparsing the xml and inserting *unit* tags or fields.