Introduction

This report will be cleaning OpenStreetMap data of Toronto, Ontario. I chose this city because I am from Mississauga, a neighbouring city of Toronto and also because this is the biggest city in Canada and I'm interested in finding and sharing interesting facts about the city.

I started by finding all the unique k values of the tags. And then I decided to look into specific key,value pairs to look for inconsistencies. I chose them randomly or the ones I wanted to look more into, or ones that are likely to have inconsistencies for example "addr:street" and "addr:province".

Problems found in the data set

Here are a couple of inconsistencies I found:

1. Street types were inconsistent

2. Province name was written differently or inaccurately multiple times

3. Same cuisine types were written in multiple ways for example, arabic cuisines were written "arabic", "arab"

I found these inconsistencies through some basic database querying and also through programming. If I found an inconsistency through querying that I hadn't found earlier, I went back and adjusted the code so the next time the xml file is exported to csv file, the changes are made and the errors are handled. The code first creates a sample file using using k=10, so every 10th element. I used a small number like 10 so I can get a better feel of what the data is like. The new sample file is called "toronto_sample.osm". The analysis I did throughout this report is based on k=10, the info in the csv files were also imported based on k=10. The "toronto_sample.osm" sample file that I submitted is k=140, to compress the file down to 10MB or under. The original osm file downloaded from openstreetmap is 1.12GB and the sample file that is k=10 is 119MB. The "cell" references in this report are talking about the code cells in toronto_analysis_code.ipynb file.

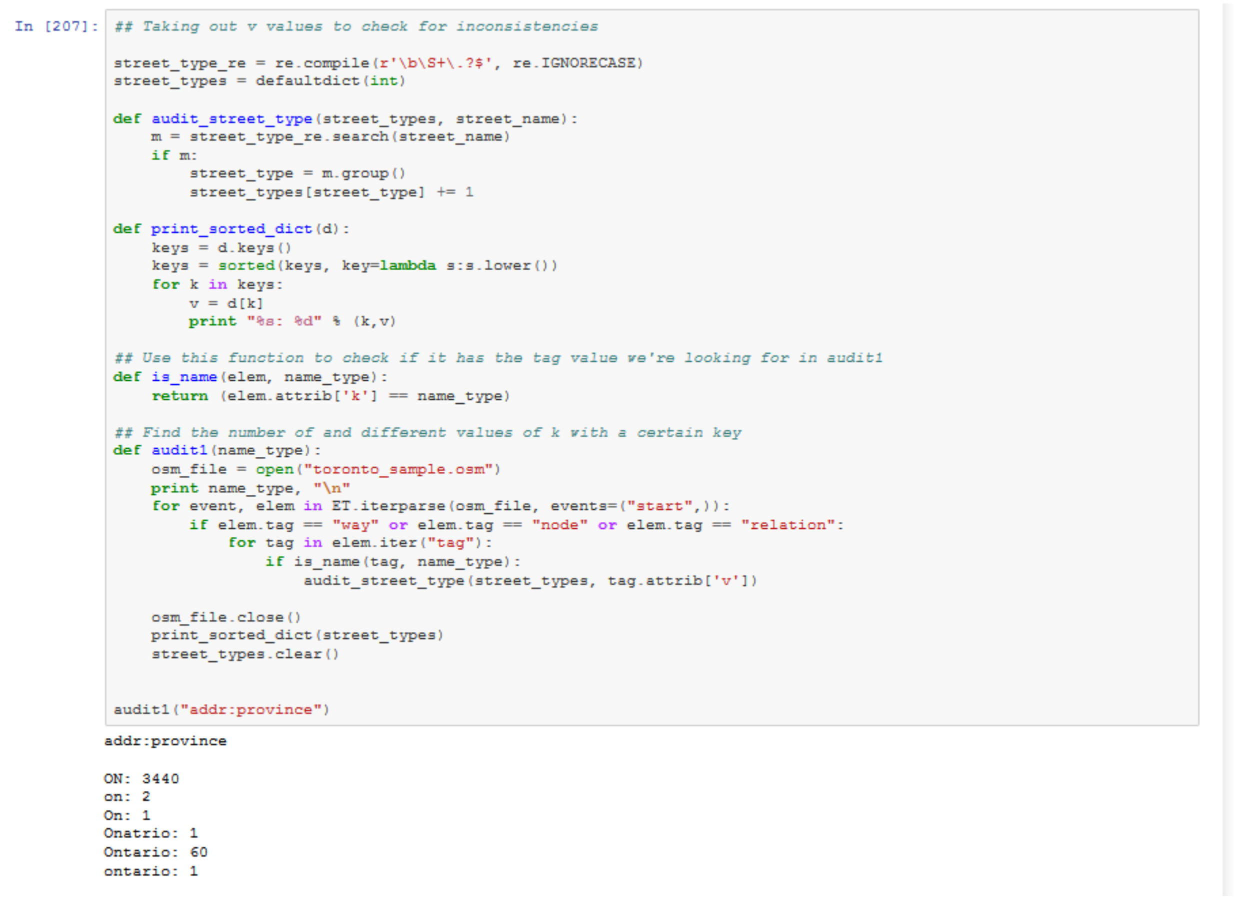
AUDITING PROCESS

The first thing I did was see all the different types of tags there are in the data like we did in the case study. The different types of tags are listed in cell [2]. By finding the different types of tags I know what places to look into and how to structure the rest of my code. I put most emphasis on the node, way and tag elements because they are likely to have errors and also they play a vital role in creating details of the map. For example, nodes are all fixed locations of a place so it is important that they are consistent. Tags reveal description and attributes of a certain place, so it is necessary to check there for errors so the map doesn't give any wrong information. Things like wrong spelling, writing one type/attribute in multiple ways can create confusion. The count_tags(filename) function below shows all the different tags/elements in the data.

Next comes the get_unique_k_values(filename) function which shows all the different k values. I found the k values to see which specific attributes I want to look into and also how different attributes are represented. For example, in the case study we looked into "addr:state" values, however in this case I had to look into "addr:province" values to find errors because in Canada the word "province" is used more than "state". I also found the unique k values to see which attributes particularly interest me. I found attributes such as different cuisines and different religious centres over Toronto to be interesting and discuss them later on in the report and find data on it through basic querying.

Dealing with province names and street names

The cell [207] finds the last word and finds the v values of a specific given k value. This is the code I used to find inconsistencies within the province names and street types. The audit1() function works by taking in as input the type of k value you are trying to find. So for example as shown in the code below, audit1("addr:province") looks for all k values that are "addr:province" and then finds its v values. It then prints out all the unique different v values and how many of them there are. If we look in the results below we see that ontario is written in number of different ways. As I predicted earlier and as we saw in the case study during the lesson, the "addr:province" tag is likely to have errors. The province names are inconsistent in this case. I will talk more about how I handle this error in the last cell. I found this code in the lessons in the data wrangling course. Although not shown below because the answer list is too large, I used audit1("addr:street") to find all the different ways the street names are written and fix any inconsistencies there. Check cell [207] in the code file to see the full code.



```
In [207]: ## Taking out v values to check for inconsistencies

          street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
          street_types = defaultdict(int)

          def audit_street_type(street_types, street_name):
              m = street_type_re.search(street_name)
              if m:
                  street_type = m.group()
                  street_types[street_type] += 1

          def print_sorted_dict(d):
              keys = d.keys()
              keys = sorted(keys, key=lambda s:s.lower())
              for k in keys:
                  v = d[k]
                  print "%s: %d" % (k,v)

          ## Use this function to check if it has the tag value we're looking for in audit1
          def is_name(elem, name_type):
              return (elem.attrib['k'] == name_type)

          ## Find the number of and different values of k with a certain key
          def audit1(name_type):
              osm_file = open("toronto_sample.osm")
              print name_type, "\n"
              for event, elem in ET.iterparse(osm_file, events=("start",)):
                  if elem.tag == "way" or elem.tag == "node" or elem.tag == "relation":
                      for tag in elem.iter("tag"):
                          if is_name(tag, name_type):
                              audit_street_type(street_types, tag.attrib['v'])

              osm_file.close()
              print_sorted_dict(street_types)
              street_types.clear()


          audit1("addr:province")
          addr:province

          ON: 3440
          on: 2
          On: 1
          Onatrio: 1
          Ontario: 60
          ontario: 1
```

The load_new_tag() function in cell [71] handles the problem for inconsistent names. I found all the different ways the province name was written in the earlier cell where I found the v values for k="addr:province". In cell 71 in the load_new_tag() function I created an array with all the inconsistent province names and then run a "if" statement, to check if the v value for the province in the xml file is in the array, and if the value in the xml file is one in the array then I convert it to "ON". This same function also handles street type errors.

CUISINE ERRORS AND OTHER DATABASE QUERIES

The load_new_tag() function also takes care of other inconsistencies I found through querying with the SQL database such as cuisine errors. The idea here is to check for the inconsistencies, adjust the code again accordingly to handle the error and then insert into the csv files again as seen fit. I created the query "SELECT key,COUNT(*) as count FROM nodes_tags GROUP BY key ORDER BY count DESC" to see all the types of k values there are. Then I executed "SELECT key,value FROM nodes_tags WHERE key='keyvalue' GROUP BY value" and then saw all the different values for that k attribute. I then skimmed over the data to find inconsistencies. If I found an inconsistency I adjusted the load_new_tag() to make changes and fix the error.

The query "SELECT *, COUNT(*) FROM nodes_tags GROUP BY value WHERE type='cuisine' " revealed the cuisine type errors. This is one part where the map can improve. For example, they label most japenese restaurants as just "japanese" however in one instance they label it as "asian;japanese". It is perhaps better to make "asian;japanese", that way it has more criteria/labels to it thus making it easier to find for example on a search engine. Cuisines of the same culture are also wrote differently in some cases. For example, Arab restaurants are written as "arab" and "arabic", Afghan restaurants as "afghan" and "afghani".

BENEFITS AND ISSUES OF CLEANING THE DATA

There is also room for improvement in this data. For starters like the querying and programming reveals there are province name errors. But before we can fix the name of the province, that is make them consistent, in this case make them "ON" for Ontario there is another small "unnecessary" tag in this data that is causing more problems and cleaning than necessary.

Benefit:

I touched on this slightly earlier but its the distinction between "state" and "province". The problem here is that there are thousands of "addr:state" tags just like there are thousands of "addr:province" tags. Because in Canada we use the word "province" it is better to just stick to province and use that tag the most. Using state sometimes and province other times can cause confusion or one can misunderstand and not clean one portion of the data. This can perhaps reduce space, improve overall consistency and be a general good guideline of where to look for what data. For example, if I wanted to find all provinces I could just search all tag values with "addr:province" and not "addr:state", "addr:province" and any other "keys" used to find the same type of value. To fix this perhaps the users creating the map can come up with a general guideline on how to pinpoint states, cities and/or any other location.

Issue:

The main issue I can think of doing this is consumption of time, it would take some time to get a team that has to find a general guideline to follow in all cases. Also if users in other countries such as the US are auditing this data, they might be looking for keys that are "addr:state" because state is used there more than province, however some querying or finding the tag values through programming can fix this problem.

Benefit #2:

The second benefit I want to discuss is the benefit of cleaning the street types. In many cases "Street" is written as "st", "west" is written as "w" and so on. Writing in shortcut isn't a problem, the problem once again is the inconsistency. If the street types are all the same, it makes it much easier to find values you are looking for and makes it easier to clean, edit and find patterns in the data.

Issue:

The issue of cleaning this data isn't a big one, however if you are changing thousands of street types to their longer version for example, "w"->"west", "st"->"street" you can take up extra additional space that is unnecessary because people can tell by looking at "w" it means west, same with most other shortcuts.

FURTHER IMPROVEMENT

To improve the data to the fullest, I think it's necessary that the team desiging the maps follow a general consistent guideline. For example, make all "st" equal to "street". Another thing to watch out for are spelling errors, businesses such as cuisines don't want to be missed out because their type was spelled wrong on the map. Or the map can have another option, that is "report" option for data analysts, everytime a data analyst sees an error they can report the error with a message on what to fix specifically. That way maps can improve in no time, or maybe if any individual finds an error they can report it as well. The data scientists can check the error for themselves first before passing it onto the team that fixes the maps.

Statistics

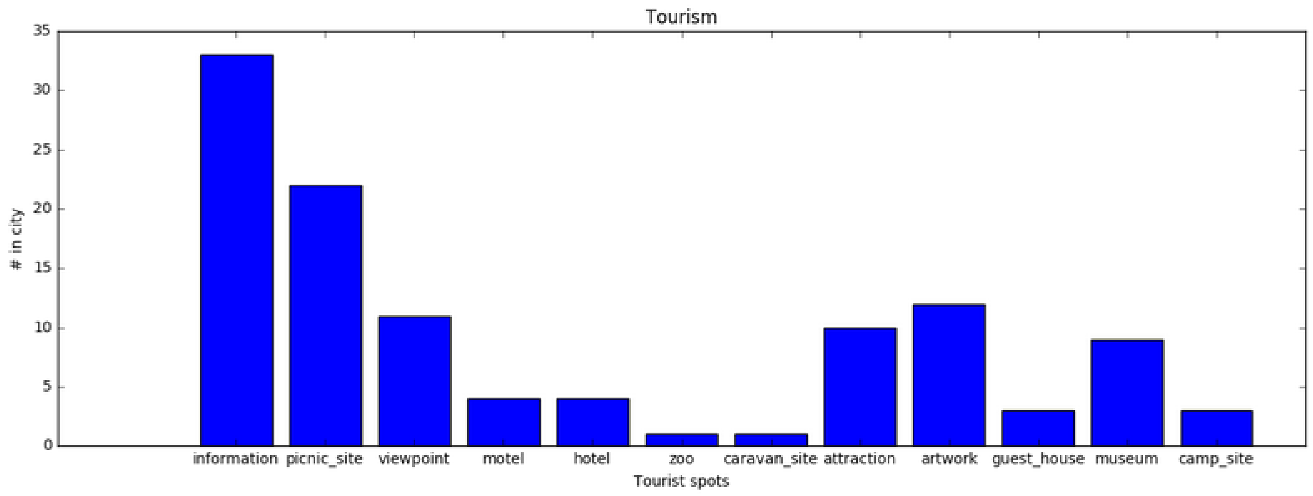
Number of unique users
1321

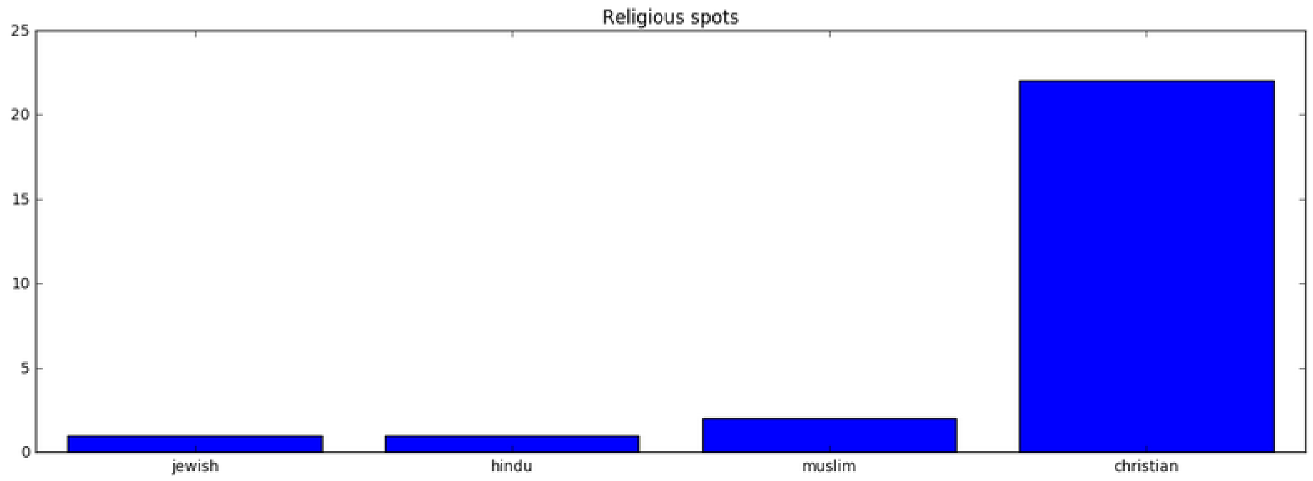
Number of nodes
501905

Number of ways
74540

```
## Number of unique users
cur.execute("SELECT COUNT(DISTINCT(uid)) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways);")
all_uid = cur.fetchall()
print "Number of unique users"
print all_uid[0][0], "\n"

cur.execute("SELECT uid, COUNT(uid) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways)")


## Number of nodes
cur.execute("SELECT COUNT(*) FROM nodes;")
all_nodes = cur.fetchall()
print "Number of nodes"
print all_nodes[0][0], "\n"

## Number of ways
cur.execute("SELECT COUNT(*) FROM ways;")
all_ways = cur.fetchall()
print "Number of ways"
print all_ways[0][0], "\n"
```

Tourism and religious spots in Toronto





CONCLUSION

Overall the data is reliable, all locations(nodes) include their lattitudes, longitudes and the tag "k" values provide sufficient information about the place. Throughout this data I focused on cleaning the "addr:street", "addr:province" and "tag k="cuisine" " portions of the xml file because I found those to be the most inconsistent. Street types were written different from one place to another, province name was written differently throughout the xml file and in some cases spelled incorrectly. I chose "addr:street" and "addr:province" because they are common to have inconsistencies as I also saw in the case study earlier when we did the data wrangling, and also because these two things are crucial for an accurate map. I also looked into other tag keys for nodes and ways such as "addr:postcode" and "addr:city" and found them to be consistent