1. Problems encountered in the map

The data of the osm project is created by humans who contribute to the data on voluntary background. The base of the data is defined by the wiki of the http://www.openstreetmap.org project. The wiki defines how to use and build data for the osm project. Since there is a lot of diffenrent data in the real world, which is hard to migrate into a standard, the data might get dirty. The map of cleveland, that is dealed with in the following project, contains a lot of data about, streetnames, shops, amenities and other elements from the real world. More than 500 unique users contributed to the data, which means that more or less approx 500 oppinions about notation of streetnames, directions or shopnames influence the map data. Especially when we take a look at the street names, the great number of contributing users has a negative effect on the notations consistency in the data. Streettypes and geographic directions are abbreviated in different ways and should be corrected. The whole data-wrangling process of the project can be summarized as:

Steps to audit the data:

- search for the last word of the street name
- check if that is an expected street type
- if it is not an expected street type, match the street type against a correct version
- Check for incorrect post-codes

Steps to correct the data:

- Replace the unexpected street types anywhere in the string, except for „St“ at the very beginning.
- Find and replace the abbreviations of directions
- Put the osm-data to a json-conform format
- Correct errors (like incorrect postcodes) at the document level in mongodb

Furthermore the osm-project has limits when it comes to calculating based on human created data of names, such as chains of supermarkets or other amenities, as will be discussed in the 3rd part of this project.

1.1 Problem encountered in street names

The street names sometimes are abbreviated. The reason for these abbreviations are different users, who have individual understandings of the street-names notations.

Search for abbreviation of street types at end of string

Often the street names are abbreviated in a popular way. The street-name „W. Market St.“ probably should be „West Market Street“. The „St.“ is located at the end of the string, so that is pretty easy to collect unknown abbreviations of street names. These abbreviations of street names can be matched against the non-abbreviated street notation and can be replaced.

Replace abbreviation of street types anywhere in the string

Some abbreviatíons are not at the end but somewhere else in the string. Whereas the use of a regular expression, that matches against the end of the whole string, can be used to collect unknown abbreviations of street names, the process of replacement has to be conducted by the

use of a regular expression that matches these street-types anywhere in the string. The name „17th St NE“ is a example where that is necessary.

Pay attention to special abbreviations
Since some street names carry a proper name, that seems to be an abbreviation but actually is not, the programmatic change is not right. An example of a programmatic notation change, that might not be correct is the change of the street name „St. Clair“ to „Street Clair“, because the „St.“ more likely is an abbreviation for „Saint“. Therefore the existence of the abbreviation „St.“ at the very beginning of the string should be checked and must not be matched against the street types.

1.2 Problem encountered in directions

Replace abbreviation of directions anywhere in the string
The street-name „17th St NE“ is likely unabbreviated „17th Street Northeast“. That example shows that not only the street type should be replaced but also the abbreviation of the direction. Since the usually used abbreviations of directions are straightforward, these can easily be searched and replaced by the unabbreviated versions.

1.3 Problem encountered in post-codes

The postcodes in the specific dataset seem to be valid except for one entry. Whereas most of the contained postcodes have values like 44xxx or a valid plus-four code 44xxx-xxxx, there is one value that is non-numeric and just „Ohio“. Since this is obviously an example of an incorrect entered value for an object it is hardly possible to correct that automatically. This problem should be considered in detail at the document level in the mongo database (used code in the „update_mongodb.txt“).

2. Overview of the data

Size of the files:
cleveland.osm : **263 MB**

cleveland.json: **287 MB**

The following part shows the performed Mongo-DB querries and the corresponding basic statistics of the cleveland area:

#number of documents:
> db.cleveland.find().count()
1279300 documents

#number of nodes:
> db.cleveland.find({"type":"node"}).count()
1164752 nodes

#number of ways:

```
> db.cleveland.find({"type":"way"}).count()
```
114548 ways

#number of unique users:
```
> db.cleveland.distinct('created.user').length
```
554 unique users

#top 1 contributing user
```
> top_user = db.cleveland.aggregate([{"$group":{"_id": "$created.user", "count":{"$sum":1}}},
                {"$sort":{"count":-1}},
                {"$limit":1}])
```
The most active User: woodpeck_fixbot is involved in 717825 changes

#Number of users appearing only once:
```
users_appearing_1time = db.cleveland.aggregate([{"$group":{"_id":"$created.user", "count":
{"$sum":1}}},
                                {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                                {"$sort":{"_id":1}},
                                {"$limit":1}])
```
There are: 86 users appearing only once

#Top 10 popular cuisines:
```
popular_cuisines = db.cleveland.aggregate([{"$match":{"amenity":{"$exists":1},
"amenity":"restaurant"}},
                            {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                            {"$sort":{"count":-1}},
                            {"$limit":10}
                            ])
```
The most popular cuisine is american and counts 28 restaurants.

3. Additional Ideas:

The OSM project consists of human created data, which leads to problems, when trying to conduct calculations, such as market-share of a specific supermarket-chain. Since a lot of people with different background and experiences contribute to the maps sometimes inconsistent data is the conseqence. As seen in the audit of the street types, where the abbreviation of street-types is not always the same, another problem can be observed in proper names of amenities. Especially when it comes to dealing with names of shop-chains or comparable institutions.

Problems with names of supermarket-chains:

Inspecting the number of supermarkets in the area of cleveland the following aggregation pipeline was used:

```
number_supermarkets = db.cleveland.aggregate([{"$match":{"shop": "supermarket"}},
                                {"$group":{"_id": "$supermarket", "count": {"$sum":1}}}])
```

The corresponding output shows that there are 107 supermarkets:

{'ok': 1.0, 'result': [{'count': 107, '_id': None}]}

Diving deeper into the supermarket-data, by inspecting the different names with the following aggregation-pipeline:

```
all_supermarkets = db.cleveland.aggregate([{"$match":{"shop": "supermarket"}},
                                           {"$group":{"_id": "$name", "count": {"$sum":1}}},
                                           {"$sort": {"count":-1}}])
```

The output shows the different names and the associated quantity of supermarkets:

All supermarket names in descending order of quantity:
[{'count': 23, '_id': 'Giant Eagle'}, {'count': 11, '_id': None}, {'count': 7, '_id': "Heinen's"}, {'count': 6, '_id': 'Walmart'}, {'count': 6, '_id': "Marc's"}, {'count': 4, '_id': 'Save-A-Lot'}, {'count': 4, '_id': "Dave's Supermarket"}, {'count': 2, '_id': "Dave's"}, {'count': 2, '_id': 'Aldi'}, {'count': 2, '_id': 'Target'}, {'count': 2, '_id': "Aldi's"}, {'count': 2, '_id': 'Acme'}, {'count': 2, '_id': "Fisher's Foods"}, {'count': 1, '_id': 'Sears Show Room'}, {'count': 1, '_id': "Zagara's Marketplace"}, {'count': 1, '_id': 'Kmart'}, {'count': 1, '_id': 'Gordon Food Service (GFS)'}, {'count': 1, '_id': 'Streetsboro Flea Market'}, {'count': 1, '_id': 'Kent International Markets'}, {'count': 1, '_id': 'Wal-Mart'}, {'count': 1, '_id': 'Acme Fresh Market'}, {'count': 1, '_id': "Heinen's Fine Foods"}, {'count': 1, '_id': 'Munchies Market'}, {'count': 1, '_id': "Buehler's Towne Market"}, {'count': 1, '_id': 'Sav-A-Lot'}, {'count': 1, '_id': "Dave's Supermarket Slavic Village"}, {'count': 1, '_id': 'Buehlers'}, {'count': 1, '_id': "Trader Joe's"}, {'count': 1, '_id': 'giant eagle'}, {'count': 1, '_id': 'KMart'}, {'count': 1, '_id': 'Kriegers'}, {'count': 1, '_id': 'Village Market'}, {'count': 1, '_id': 'Marcs Supermarket'}, {'count': 1, '_id': 'Marcs Grocer'}, {'count': 1, '_id': 'Whole Foods'}, {'count': 1, '_id': 'The Fresh Market'}, {'count': 1, '_id': 'Dollar General'}, {'count': 1, '_id': 'Walmart Supercenter'}, {'count': 1, '_id': 'Acme No. 1'}, {'count': 1, '_id': 'Newton Falls IGA'}, {'count': 1, '_id': 'Freshway'}, {'count': 1, '_id': 'Discount Drug Mart'}, {'count': 1, '_id': 'Mentor Family Foods'}, {'count': 1, '_id': 'Save a Lot'}, {'count': 1, '_id': 'Buehlers Fresh Foods '}, {'count': 1, '_id': 'Whole Foods Market'}, {'count': 1, '_id': 'Miles Farmers Market'}]

At this stage the calculation of market share for „Walmart“-stores, at position 4, with 6 branches in the area would lead to 6/107 = 5.6 percent.

By visually inspecting the output it becomes clear that some supermarkets occur under different names. For Example „Walmart“, „ACME“ and „Aldi“ occur at least twice. The problem with that data is, that it is not possible to do accurate calculations on the data.

To get a feeling for the occurence of different names i wrote a function (can be found in „aditional_idea_supermarket.py“) to inspect the names.

The function uses regular expressions to check for occurences of „walmart“, „acme“ or „aldi“-like names. The output of the function is a default dictionary set with different names for the same supermarket.

```
defaultdict(<class 'set'>,
{re.compile('(wal)', re.IGNORECASE): {'Wal-Mart', 'Walmart', 'Walmart Supercenter'},
re.compile('(aldi)', re.IGNORECASE): {"Aldi's", 'Aldi'},
re.compile('acme', re.IGNORECASE): {'Acme', 'Acme Fresh Market', 'Acme No. 1'}})
```

As can be seen in the above printed dictionary „nameset“, for example three different names of Walmart or ACME Supermarket are used.

Possible reason for the different names:

I assume that the people who created the data, wanted to give some information about the details of the branch-size or the existing products.

Whats the problem with the data:

Using different names for the same shop-chain makes proper calculations complex. Either the data has to be cleaned or the rise of that kind of error has to be avoided.

Downside of automated cleaning the names:

A script like the „audit_supermarket_name“ from the „inspect_supermarkets.py“ could be the base of a data cleaning process. The cleaning could be realised like the cleaning in the audit.py file. By comparing the name of the supermarket with a mapping-directory and renaming the supermarket if a "normalised" name exists.

The cleaning has to be done very carefully, because a fully automated data-cleaning can lead to other problems. For expample: It might be possible, that a supermarket is named by its owner who is called „acme“. The cleaning of the 'Acme Fresh Market' to a standardised name like „acme“ would not be the correct way of cleaning the data, because there was no problem with the name in the first place.

Conclusion:

After accounting the other walmart-branches with different names, the calculation of market-share for walmart has to be corrected by two branches which adds market-share from 5.61 percent to 7.48 percent.

Since the names of supermarket chains may lead to inconsistent data and the automated cleaning is not the best solution, the additional information like „supercenter“ or „fresh market“ might be usefully stored in another additional field of the document. Since the information about the size of the supermarket is already documented in the wiki http://wiki.openstreetmap.org/wiki/Tag:shop%3Dsupermarket under the point „similar tags“, that information might not properly communicated to the user.

The norm for the names should be communicated thoroughly to those who create the data. It might be possible that the data has to be confirmed by some admin or very active user, who is aware of the name-inconsistency problems and who is able to decide which name is the right one.